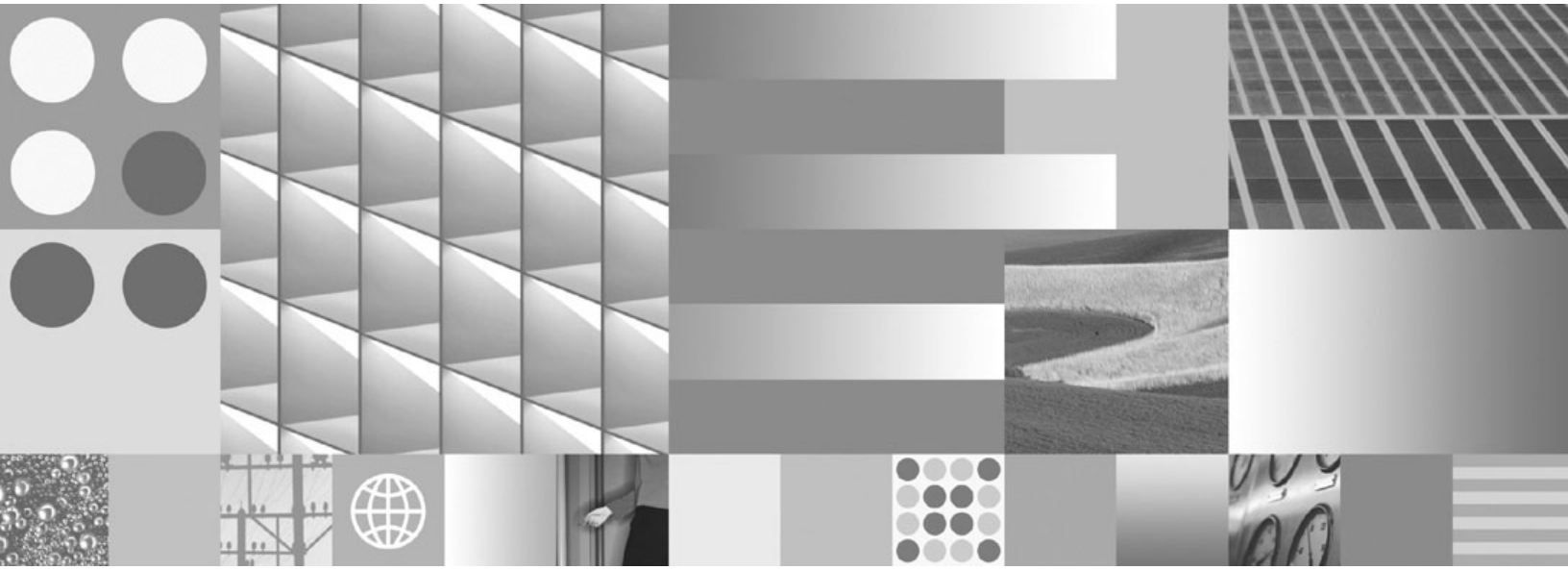
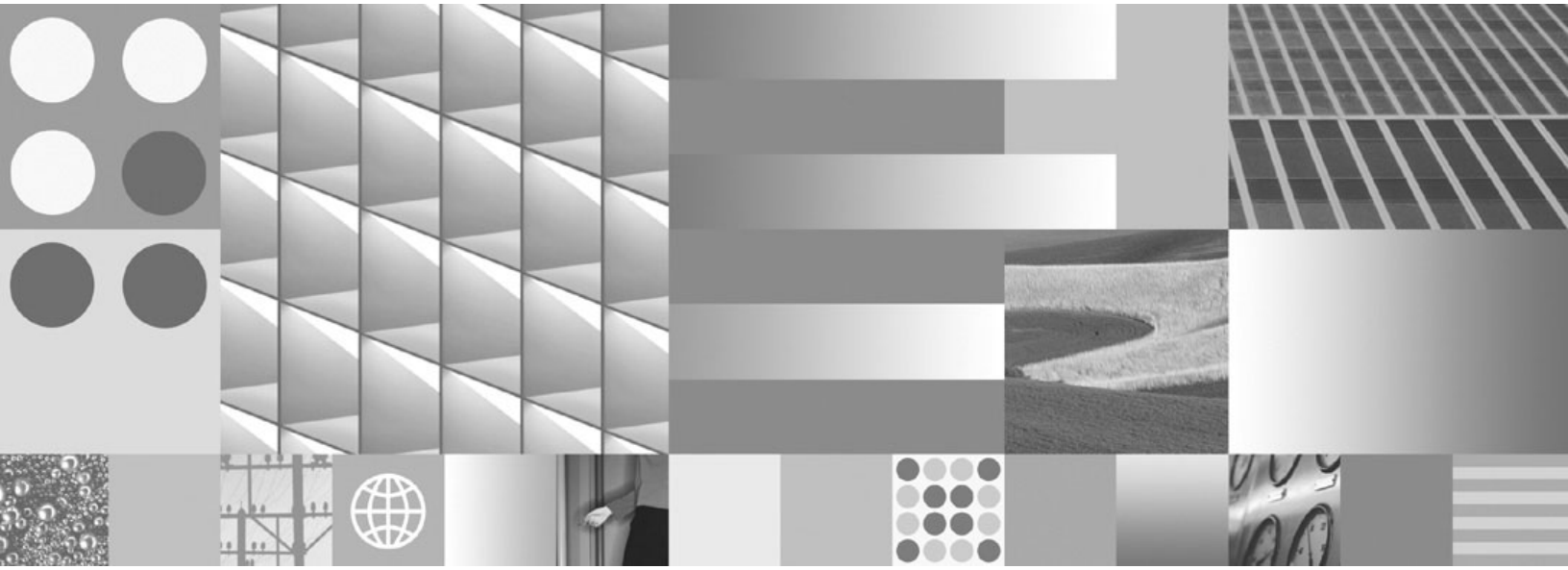
IBM DB2 9.7
for Linux, UNIX, and Windows



データベースのモニタリング　ガイドおよびリファレンス

SC88-5872-00
(英文原典：SC27-2458-00)

IBM DB2 9.7
for Linux, UNIX, and Windows



データベースのモニタリング　ガイドおよびリファレンス

SC88-5872-00
(英文原典：SC27-2458-00)

ご注意

本書および本書で紹介する製品をご使用になる前に、 863 ページの『付録 B. 特記事項』に記載されている情報をお読みください。

IBM 資料は、オンラインでご注文いただくことも、ご自分の国または地域の IBM 担当員を通してお求めいただくこともできます。

- オンラインで資料を注文するには、www.ibm.com/shop/publications/order にある IBM Publications Center をご利用ください。
- ご自分の国または地域の IBM 担当員を見つけるには、www.ibm.com/planetwide にある IBM Directory of Worldwide Contacts をお調べください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： SC27-2458-00
IBM DB2 9.7
for Linux, UNIX, and Windows
Database Monitoring Guide and Reference

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2009.7

目次

本書について

「システム・モニター *ガイドおよびリファレンス*」では、データベースおよびデータベース・マネージャーに関するさまざまな種類の情報を収集する方法について説明しています。

本書では、収集した情報を使用してデータベース・アクティビティーを理解したり、パフォーマンスを向上させたり、問題の原因を判別したりする方法についても説明しています。

第 1 部 インターフェースのモニター

第 1 章 データベース・モニター

データベースのモニターは、データベース管理システムのパフォーマンスと正常性を保守するためにきわめて重要なアクティビティーです。モニターを容易にするために、DB2® はデータベース・マネージャー、データベース、および他の接続されているアプリケーションから情報を収集します。この情報を使用して、例えば以下の事柄を行うことができます。

- データベース使用パターンに基づいたハードウェア要件の予測
- 個々のアプリケーションまたは SQL 照会のパフォーマンスの分析
- 索引および表の使用量の追跡
- システム・パフォーマンス低下の原因の正確な指摘
- 最適化アクティビティー (データベース・マネージャー構成パラメーターの変更、索引の追加、SQL 照会の変更など) の影響の評価

第 2 章 モニター表関数の概要

DB2 バージョン 9.7 からは、従来のシステム・モニターに代わる軽量の代替手段により、モニター・データにアクセスできるようになりました。モニター表関数を使用して、システム、アクティビティー、またはデータ・オブジェクトのデータを収集および表示します。

モニター・エレメントのデータは、メモリー内に継続的に累積され、照会に使用することができます。単一オブジェクト (例えば、サービス・クラス A または表 TABLE1) についてのデータまたはすべてのオブジェクトについてのデータを受け取ることを選択できます。

データベース・パーティション環境内でこれらの表関数を使用すると、単一のパーティションについてのデータまたはすべてのパーティションについてのデータを受け取ることを選択できます。すべてのパーティションのデータを受け取ることを選択した場合、表関数は各パーティションについて 1 行を返します。 SQL を使用すると、すべてのパーティションの値を合計して、すべてのパーティションのモニター・エレメントの値を取得することができます。

表関数を使用したシステム情報のモニター

システム・モニター・パースペクティブは、アプリケーション要求を処理するためにデータ・サーバーによって費やされた作業量および労力全体を対象としています。このパースペクティブから、データ・サーバーの実行内容を全体として、およびアプリケーション要求の特定のサブセットごとに判別することができます。

要求モニター・エレメントと呼ばれる、このパースペクティブのモニター・エレメントは、要求の処理と関連するデータ・サーバー操作の全範囲を対象とします。

要求モニター・エレメントはメモリー内に継続的に累積されて集約されるため、それらは即時に照会に使用可能です。要求モニター・エレメントは、ワークロード管理 (WLM) オブジェクト階層のさまざまなレベル (作業単位別、ワークロード別、サービス・クラス別) の要求全体について集約されます。それらは接続別でも集約されます。

以下の表関数を使用して、現在のシステム・モニター情報にアクセスします。

- MON_GET_SERVICE_SUBCLASS および MON_GET_SERVICE_SUBCLASS_DETAILS
- MON_GET_WORKLOAD および MON_GET_WORKLOAD_DETAILS
- MON_GET_CONNECTION および MON_GET_CONNECTION_DETAILS
- MON_GET_UNIT_OF_WORK および MON_GET_UNIT_OF_WORK_DETAILS

この一連の表関数により、特定の集約レベルの要求モニター・エレメントをドリルダウンしたり、それに焦点を当てたりすることができます。表関数は対で提供されています。1 つは一般に使用されるデータへのリレーショナル・アクセス用、もう 1 つは使用可能なモニター・エレメントの完全セットへの XML アクセス用です。

システム・モニター情報は、これらの表関数により、デフォルトでは新規データベースを対象に収集されます。デフォルト設定は、以下のいずれかまたは両方の設定を使用して変更できます。

- データベース構成パラメーター **mon_req_metrics** は、すべてのサービス・クラスにおける最小レベルの収集を指定します。
- CREATE/ALTER SERVICE CLASS ステートメントの COLLECT REQUEST METRICS 節は、サービス・スーパークラスの収集のレベルを指定します。この設定を使用して、すべてのサービス・クラスに設定されている最小レベルの収集をオーバーライドして、特定のサービス・クラスの収集レベルを上げることができます。

各設定に指定可能な値は、以下のとおりです。

None 要求モニター・エレメントは収集されません。

BASE すべての要求モニター・エレメントが収集されます。

例えば、サービス・クラスのサブセットのみのシステム・モニター情報を収集するには、以下を実行します。

1. データベース構成パラメーター **mon_req_metrics** を NONE に設定します。
2. 対象とする各サービス・クラスについて、CREATE/ALTER SERVICE CLASS ステートメントの COLLECT REQUEST METRICS 節を BASE に設定します。

表関数を使用したアクティビティーのモニター

アクティビティー・モニター・パースペクティブは、アクティビティーの実行に関係した、データ・サーバー処理のサブセットに焦点を当てています。 SQL ステートメントのコンテキストでは、「アクティビティー」という語は、SQL ステートメントのセクションの実行を表します。

アクティビティー・モニター・エレメントと呼ばれる、このパースペクティブのモニター・エレメントは、要求モニター・エレメントのサブセットです。アクティビティー・モニター・エレメントは、ステートメント・セクションの実行で行われた作業の各局面を測定します。アクティビティー・モニターには、アクティビティーの SQL ステートメント・テキストなどの他の情報も含まれます。

進行中のアクティビティーの場合、アクティビティー・メトリックはメモリーに累積されます。 SQL ステートメントのアクティビティーの場合、アクティビティー・メトリックはパッケージ・キャッシュにも累積します。パッケージ・キャッシュには、各 SQL ステートメント・セクションのすべての実行に対するアクティビティー・メトリックが集約されます。

以下の表関数を使用して、アクティビティーの現行データにアクセスします。

MON_GET_ACTIVITY_DETAILS
表関数が呼び出されるときに、進行中の個々のアクティビティーに関するデータを返します。データは XML 形式で返されます。

MON_GET_PKG_CACHE_STMT
セクションのすべての実行について集約された、個々の SQL ステートメント・セクションのデータを返します。データはリレーショナル形式で返されます。

アクティビティー・モニター情報は、デフォルトでは新規データベースを対象に収集されます。デフォルト設定は、以下のいずれかまたは両方の設定を使用して変更できます。

- **mon_act_metrics** データベース構成パラメーターは、すべてのワークロードにおける最小レベルの収集を指定します。
- CREATE/ALTER WORKLOAD ステートメントの COLLECT ACTIVITY METRICS 節は、すべてのワークロードに設定される最小レベルの収集をオーバーライドして、特定のワークロードの収集レベルを指定します。

各設定に指定可能な値は、以下のとおりです。

None アクティビティー・モニター・エレメントは収集されません。

BASE すべてのアクティビティー・モニター・エレメントが収集されます。

例えば、選択したワークロードのみのアクティビティー・モニター・エレメントを収集するには、以下を実行します。

1. **mon_act_metrics** データベース構成パラメーターを NONE に設定します。
2. CREATE/ALTER WORKLOAD ステートメントの COLLECT ACTIVITY METRICS 節を BASE に設定します。デフォルトでは、その他のワークロードの値は NONE です。

表関数を使用したデータ・オブジェクトのモニター

データ・オブジェクト・モニター・パースペクティブは、表、索引、バッファー・プール、表スペース、およびコンテナーなどのデータ・オブジェクトに対して実行された操作に関する情報を提供します。

各オブジェクト・タイプに対してさまざまなモニター・エレメントのセットが使用できます。データ・オブジェクトのモニター・エレメントは、要求が対象オブジェクトの処理に関与するたびに増分されます。例えば、特定の表からの行の読み取りに関与する要求を処理する場合、その表の行読み取りのメトリックが増分されます。

以下の表関数を使用して、データ・オブジェクトの現行の詳細にアクセスします。

- MON_GET_BUFFERPOOL
- MON_GET_TABLESPACE
- MON_GET_CONTAINER
- MON_GET_TABLE
- MON_GET_INDEX

これらの表関数は、リレーショナル形式でデータを返します。

データ・オブジェクトの履歴データにはアクセスできません。

データ・オブジェクト・モニター・エレメントは、デフォルトでは新規データベースを対象に収集されます。 **mon_obj_metrics** データベース構成パラメーターを使用して、表関数により収集されるデータの量を削減することができます。

この構成パラメーターに指定可能な値は、以下のとおりです。

None データ・オブジェクト・モニター・エレメントは収集されません。

BASE すべてのデータ・オブジェクト・モニター・エレメントが収集されます。

mon_obj_metrics パラメーターの設定対象に関係なく、データは以下の表関数により報告されるモニター・エレメントについては常に収集されます。

- MON_GET_TABLE
- MON_GET_INDEX

以下の表関数により報告されるデータ・オブジェクト・モニター・エレメントの収集を停止するには、**mon_obj_metrics** 構成パラメーターを NONE に設定します。

- MON_GET_BUFFERPOOL
- MON_GET_TABLESPACE
- MON_GET_CONTAINER

第 3 章 イベント・モニター

イベント・モニターは、 CREATE EVENT MONITOR ステートメントで指定されたイベント・タイプに関する情報を戻します。それぞれのイベント・タイプごとに、特定の時点でモニター情報が収集されます。

以下の表は、使用可能なイベント・タイプ、モニター・データが収集される時点、およびそれぞれのイベント・タイプについて入手できる情報をリストしています。最初の列にある使用可能なイベント・タイプは、 CREATE EVENT MONITOR ステートメントで使用されるキーワードに対応しています。ここでイベント・タイプが定義されます。

データが発生するように定義されたイベントに加えて、 FLUSH EVENT MONITOR SQL ステートメントを使用してイベントを生成することもできます。この方式によって生成されたイベントは、フラッシュされたイベント・モニターに関連したすべてのモニター・タイプ (DEADLOCKS および DEADLOCKS WITH DETAILS を除く) に関する現行のデータベース・モニター値とともに書き込まれます。

ステートメント・イベント・モニターを使用して SQL プロシージャーの実行をモニターする際には、以下のようになります。

- INSERT、SELECT、DELETE、UPDATE などのデータ操作言語 (DML) ステートメントは、イベントを生成します。
- 変数割り当てや制御構造 (例えば WHILE や IF) などのプロシージャー・ステートメントは、決定論的にイベントを生成しません。

表 1. イベント・タイプ

イベント・タイプ	データが収集される時点	入手できる情報
DEADLOCKS[1]	デッドロック検出時	関係しているアプリケーション、および競合しているロック。
DEADLOCKS WITH DETAILS[1]	デッドロック検出時	関係するアプリケーションについての広範囲の情報。関係するステートメント (およびステートメント・テキスト) の識別や保持されているロックなど。 DEADLOCKS イベント・モニターではなく DEADLOCKS WITH DETAILS イベント・モニターを使用すると追加の情報が収集されるため、デッドロックが発生したときにパフォーマンス・コストの負担になります。

表 1. イベント・タイプ (続き)

イベント・タイプ	データが収集される時点	入手できる情報
DEADLOCKS WITH DETAILS HISTORY[1]	デッドロック検出時	DEADLOCKS WITH DETAILS イベント・モニターで報告されるすべての情報。ならびにデータベース・パーティションのデッドロック・シナリオにかかわるロック (このデータベース・パーティションで保持されるロック) を所有する、各アプリケーションの現行作業単位のステートメント履歴。 DEADLOCKS WITH DETAILS HISTORY イベント・モニターを使用すると、ステートメント履歴を追跡することになるので、活動化したときに若干モニター・パフォーマンスの負担になります。
DEADLOCKS WITH DETAILS HISTORY VALUES[1]	デッドロック検出時	デッドロックの詳細およびデッドロックの詳細履歴で報告されるすべての情報。ならびにステートメントの実行時にパラメーター・マーカーに提供されるあらゆる値。 DEADLOCKS WITH DETAILS HISTORY VALUES イベント・モニターを使用すると、さらにデータ値をコピーすることになるため、活動化したときにパフォーマンス・コストの大きな負担になります。
STATEMENTS	SQL ステートメント終了時	ステートメントの開始/停止時刻、使用されている CPU、動的 SQL のテキスト、SQLCA (SQL ステートメントの戻りコード)、 およびその他のメトリック (フェッチ・カウントなど)。 **注:** ステートメントの開始/停止時刻は、TIMESTAMP スイッチが OFF のときは使用できません。
	サブセクション終了時	パーティション・データベースの場合、使用されている CPU、実行時間、表、および表キューに関する情報。
TRANSACTIONS[2]	作業単位の終了時	作業単位の作業開始/停止時刻、直前の作業単位時間、使用されている CPU、ロッキング、およびロギングのメトリック。 XA で実行している場合は、トランザクション・レコードは生成されません。
CONNECTIONS	接続終了時	すべてのアプリケーション・レベルのカウンター。
DATABASE	データベース非活動化時	すべてのデータベース・レベルのカウンター。
BUFFERPOOLS	データベース非活動化時	バッファー・プール、プリフェッチャー、ページ・クリーナー、および個々のバッファー・プールの直接 I/O のカウンター。
TABLESPACES	データベース非活動化時	バッファー・プール、プリフェッチャー、ページ・クリーナー、および個々の表スペースの直接 I/O のカウンター。
TABLES	データベース非活動化時	個々の表の、読み取り/書き込みが行われる行。

表 1. イベント・タイプ (続き)

イベント・タイプ	データが収集される時点	入手できる情報
アクティビティー	COLLECT ACTIVITY DATA オプションがオンになっているサービス・クラス、ワークロード、または作業クラスで実行したアクティビティーの完了時。WLM_CAPTURE_ACTIVITY_IN_PROGRESS ストアード・プロシージャーを実行しているその時点のターゲット・アクティビティーに関するデータも収集されます。 COLLECT ACTIVITY DATA オプションが有効になっているアクティビティーがしきい値に違反する場合も、データが収集されます。	アクティビティー・レベル・データ。COLLECT ACTIVITY DATA の一部として WITH DETAILS を指定した場合には、このオプションの対象アクティビティーに関するステートメントとコンパイル環境の情報が組み込まれます。AND VALUES も指定した場合は、このオプションの対象アクティビティーの入力データ値も組み込まれます。
統計	*period* 分ごと。*period* は統計の収集間隔の時間の長さです。この期間は、WLM_COLLECT_INT データベース構成パラメーターで定義されます。 WLM_COLLECT_STATS ストアード・プロシージャーが呼び出される際にもデータが収集されます。	システム上の個々のサービス・クラス、ワークロード、または作業クラス中で実行されたアクティビティーから統計が計算されます。
しきい値違反	しきい値違反の検出時。	しきい値違反情報。
ロッキング	ロック・タイムアウト、デッドロック、指定期間を超えたロック待機のいずれかのイベント・タイプ (構成設定による) の検出時。	ロック・イベント・レコード。
作業単位	作業単位の完了時	作業単位イベント・レコード。レコードに要求メトリックを含めるオプション。

[1] このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

[2] このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR UNIT OF WORK ステートメントを使用して、トランザクション・イベントをモニターしてください。

注: 「デッドロックの詳細」イベント・モニターが、新規に作成されるデータベースごとに作成されます。このイベント・モニターは DB2DETAILDEADLOCK という名前で、データベースが活動化されると開始され、データベース・ディレクトリー内のファイルに書き込みます。このイベント・モニターによるオーバーヘッドは、これをドロップすることによって回避することができます。DB2DETAILDEADLOCK イベント・モニターは使用すべきではありません。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

未フォーマット・イベント表に書き込むイベント・モニター

DB2 9.7 では、未フォーマット・イベント表をターゲット・タイプとするいくつかのイベント・モニターが加えられました。このタイプのイベント・モニターは、パフォーマンスに優れ、新規 CREATE EVENT MONITOR ステートメント・オプション、および分析用データにアクセスするための新規インターフェースを提供します。

未フォーマット・イベント表のイベント・モニターの特性は、以下のタスクの実行方法に影響を与えます。

- イベント・モニターの作成およびデータ収集の構成
- イベント・モニター操作の管理
- イベント・モニターによりキャプチャーされたイベント・データへのアクセス

一般に、モニターのニーズはすべて、特定タイプのイベント (例えばロック・イベント) について、データベースごとに単一のイベント・モニターを作成することで満たすことができます。設定を変更することによって、モニターで収集可能なデータの量を増減させて、変化するモニターのニーズに対応することができます。これは、複数のイベント・モニターを作成し、それぞれを特定のモニター・ニーズを収集するために適合させることが一般的な方法であった、旧式のイベント・モニターとは対照的です。

イベント・モニターと関連した未フォーマット・イベント表の作成

イベント・モニターの作成における 1 つの面は、モニターが収集するデータを書き込む場所を指定することです。このタイプのイベント・モニターは、バイナリー・フォーマットのデータを、常に未フォーマット・イベント表に書き込みます。未フォーマット・イベント表は、DB2 9.7 で導入されたターゲット・タイプです。未フォーマット・イベント表は、イベント・モニターを作成するたびに暗黙に作成されます。このタイプのイベント・モニターの CREATE EVENT ステートメントには、WRITE TO UNFORMATTED EVENT TABLE 節が含まれます。

CREATE EVENT MONITOR ステートメントには、未フォーマット・イベント表を構成するための以下のオプションが組み込まれています。

- 表名 - デフォルトでは、未フォーマット・イベント表は、イベント・モニター名に基づいて名前指定されます。
- 表スペース名 - デフォルトでは、フォーマットされていない表は、ユーザーが USE 特権を持っている場合その対象となる表スペース IBMDEFAULTGROUP に作成されます。ただし、推奨されている構成方法は、以下で説明するとおり、イベント・モニターに最適化された表スペースを定義することです。
- PCTDEACTIVATE - デフォルト値は 100 です。これは表スペースが満杯になると、イベント・モニターが非活動化されることを意味します。

未フォーマット・イベント表の表スペースに関する以下の考慮事項を検討する必要があります。

- イベント・モニターの未フォーマット・イベント表の表スペースを作成して、パフォーマンス用に構成します。以下の節を CREATE TABLESPACE ステートメントで使用します。

– 可能な限り大きいページ・サイズ (PAGESIZE) を指定します (最大で 32KB まで)。
– NO FILE CACHING SYSTEM オプションを指定します。

- パーティション・データベース環境では、表スペースが存在するパーティションを考慮します。ターゲットの未フォーマット・イベント表の表スペースが、あるデータベース・パーティション上に存在していない場合、そのターゲットの未フォーマット・イベント表のデータは無視されます。この動作を利用すると、ユーザーは、特定のデータベース・パーティションにのみ存在する表スペースを作成することにより、選択するモニター用にデータベース・パーティションのサブセットを選択できることになります。

未フォーマット・イベント表に関する他の役立つ情報を、以下に示します。

- SYSCAT.EVENTTABLES カタログ・ビューは、イベント・モニター、関連した未フォーマット表、および他の詳細をリストします。
- 未フォーマット・イベント表の列については、関連リンクにリストされているトピックで説明されています。

イベント・モニターのデータ収集の構成

イベント・モニターのセットアップには、収集するデータの指定が関係しています。モニターするシステム・ワークロードのサブセット、収集するイベントのタイプ、各イベントについて収集する詳細の程度、およびデータ・キャプチャーの使用可能化/使用不可化 (データ・キャプチャーのオン/オフ) など、いくつかの面があります。データ収集の構成の考慮事項は、以下のとおりです。

- このタイプのイベント・モニターでは、CREATE/ALTER WORKLOAD ステートメントを使用して個々のワークロード定義のプロパティーを設定することにより、まずデータ収集を構成します。つまり、さまざまなワークロードに対してさまざまなデータ収集設定を指定することができます。 CREATE/ALTER WORKLOAD ステートメントには、イベント・モニターの特定のタイプに固有の節が含まれています。
- デフォルトでは、このタイプのイベント・モニターが自動的にアクティブになります。CREATE EVENT ステートメントに MANUALSTART キーワードを指定することで、イベント・モニターを手動でアクティブにするように指定できます。次いで、イベント・モニターを SET EVENT MONITOR STATE ステートメントで制御できます。
- 別の文脈で言及したとおり、パーティション・データベース環境では、イベント・モニターを使用してモニターするデータベース・パーティションのサブセットを選択できます。イベント・モニターの作成時に、モニターするパーティション上のみに存在する未フォーマット・イベント表の表スペースを指定します。未フォーマット・イベント表が特定のデータベース・パーティション上に存在しない場合、イベント・モニターはそのパーティションのデータを収集しません。
- このタイプのイベント・モニターのデータ収集は、UPDATE MONITOR SWITCHES ステートメントを使用して設定されたシステム・モニター・スイッチ設定によって影響を受けることはなく、イベント・キャプチャーが SET EVENT MONITOR ステートメントを使用してオン/オフにされることもありません。

イベント・モニター操作の管理

以下は、イベント・モニターの実行中の操作を管理するためのガイダンスとなります。

- 収集するデータの指定は、ALTER WORKLOAD ステートメントを使用することによって、いつでも変更できます。
- CREATE EVENT ステートメントで MANUALSTART オプションを指定した場合、SET EVENT MONITOR STATE ステートメントを使用してデータ収集を開始および停止することができます。
- 未フォーマット・イベント表は、手動で整理する必要があります。
- 割り当てられている最大スペースに未フォーマット・イベント表が達した場合、イベント・モニターは非非アクティブ化されます。
- イベント・モニターが不要になった場合、DROP ステートメントを使用してイベント・モニターをドロップします。 DROP ステートメントを発行しても、イベント・モニターに関連付けられている未フォーマット・イベント表はドロップされません。関連する未フォーマット・イベント表は、イベント・モニターがドロップされた後に、手動でドロップする必要があります。未フォーマット・イベント表をドロップしない場合、未フォーマット・イベント表の名前が既存のものと同じ別のイベント・モニターをその後作成しようとするときに、問題が生じる場合があります。

イベント・モニターによりキャプチャーされたイベント・データへのアクセス

このタイプのイベント・モニターは、バイナリー・フォーマットのデータを、未フォーマット・イベント表に書き込みます。このデータには、db2evmonfmt コマンドまたはこの目的で提供されているルーチンを使用してアクセスできます。

db2evmonfmt コマンドを使用すると、以下を実行できます。

- イベント ID、イベント・タイプ、時間枠、アプリケーション、ワークロード、またはサービス・クラスの各属性に基づいて、関心対象のイベントを選択します。
- テキスト・レポートまたはフォーマット済み XML 文書のどちらの形式で出力を受け取るかを選択します。
- db2evmonfmt によって提供されているものを使用する代わりに、独自の XSLT スタイル・シートを作成して、出力形式を完全に制御します。

さらに、以下のルーチンを使用して、未フォーマット・イベント表からデータを抽出することもできます。

- EVMON_FORMAT_UE_TO_XML - 未フォーマット・イベント表から XML 文書にデータを抽出します。
- EVMON_FORMAT_UE_TO_TABLES - 未フォーマット・イベント表から一連のリレーショナル表にデータを抽出します。

これらのルーチンにより、SELECT ステートメントを使用して、未フォーマット・イベント表からの抽出対象とする行を厳密に指定できます。

未フォーマット・イベント表列の定義

WRITE TO UNFORMATTED EVENT TABLE 節を含む CREATE EVENT ステートメントを発行すると、フォーマットされていないイベント表が作成されます。列定義は、分析するデータを抽出する場合や、表で不要なデータを整理する場合に役立ちます。

フォーマットされていないイベント表の列定義は、以下のいずれかのルーチンを使用して、フォーマットされていないイベント表からデータを抽出する場合に役立ちます。

- EVMON_FORMAT_UE_TO_XML - 未フォーマット・イベント表から XML 文書にデータを抽出します。
- EVMON_FORMAT_UE_TO_TABLES - 未フォーマット・イベント表から一連のリレーショナル表にデータを抽出します。

これらのルーチンの呼び出しは、抽出する行を指定する SELECT ステートメントを受け入れます。フォーマットされていないイベント表列定義を使用して、SELECT ステートメントの構成を支援します。

フォーマットされていないイベント表に書き込まれたイベント・データの自動パージは行われません。データは表から手動でパージする必要があります。フォーマットされていないイベント表の列定義は、ターゲットとなるレコードのセットをパージする場合に役立ちます。別のオプションは、TRUNCATE TABLE コマンドを使用してすべての表の行を除去することです。

CREATE EVENT MONITOR ステートメントの一部として、関連した未フォーマット・イベント表に付ける名前を指定できます。指定されない場合、イベント・モニターと同じ名前がデフォルトで使用されます。SYSCAT.EVENTTABLES カタログ・ビューは、イベント・モニター、関連した未フォーマット表、および他の詳細をリストします。

以下の表は、フォーマットされていないイベント表の列について記述したものです。キー列は event_data 列です。他の列は、関心があるイベントを探索するために使用できる ID を表します。表列の他の属性については、DESCRIBE ステートメントを発行してください。

表 2. 未フォーマット・イベント表列の定義

列名	列データ・タイプ	列の記述
appl_id	VARCHAR	イベントが発生したアプリケーションの ID。 NULL 値は、アプリケーション ID が使用不可だったことを示します。
appl_name	VARCHAR	イベントが発生したアプリケーションの名前。 NULL 値は、アプリケーション名が使用不可だったことを示します。

表 2. 未フォーマット・イベント表列の定義 (続き)

列名	列データ・タイプ	列の記述
client_acctng	VARCHAR	このイベント用の CLIENT_ACCTNG 特殊レジスターの現行値。NULL 値は、クライアント・アカウンティングが使用不可だったことを示します。
client_applname	VARCHAR	このイベント用の CLIENT_APPLNAME 特殊レジスターの現行値。NULL 値は、クライアント・アプリケーションの名前が使用不可だったことを示します。
client_userid	VARCHAR	このイベント用の CLIENT_USERID 特殊レジスターの現行値。NULL 値は、クライアントのユーザー ID が使用不可だったことを示します。
client_wrkstnname	VARCHAR	このイベント用の CLIENT_WRKSTNNAME 特殊レジスターの現行値。NULL 値は、クライアント・ワークステーションの名前が使用不可だったことを示します。
event_correlation_id	BIT DATA	オプションのイベント相関 ID。NULL 値は、イベント相関 ID が使用不可だったことを示します。 この値はイベント・モニター・タイプに基づいています。 • LOCKING - 将来の利用のために予約済み • UOW- 将来の利用のために予約済み
event_data	BLOB	イベント・モニターによってキャプチャーされるイベント用のイベント・レコード・データ全体 (元のバイナリー形式で保管される)。

表 2. 未フォーマット・イベント表列の定義 (続き)

列名	列データ・タイプ	列の記述
event_id	INTEGER	ロック・イベント・モニター・レコードの場合、データベース全体でユニークなイベント ID。 ID はデータベースの活動化時にリサイクルされます。**event_timestamp**、**event_id**、**member**、および **event_type** の組み合わせによってユニーク性が保証されます。 UOW イベント・モニター・レコードの場合、接続ごとにユニークな UOW ID の別名。 **event_timestamp**、**event_id**、**member**、**event_type** および **appl_id** の組み合わせによってユニーク性が保証されます。
event_timestamp	TIMESTAMP	イベントがイベント・モニターによって生成されたタイム・スタンプ。すべての子レコードは、親レコードと同じタイム・スタンプを共有します。
event_type	VARCHAR	検出のメンバーで発生したイベント・タイプ。
member	SMALLINT	イベントが発生したメンバー。
partitioning_key	INTEGER	表のパーティション・キー。これは、イベント・モニターが実行されるデータベース・パーティションで、挿入操作がローカルに実行されるためのものです。
record_seq_num	INTEGER	event_data 列内に保管されるレコードのシーケンス番号。
record_type	INTEGER	event_data 列内に保管されるレコードのタイプ。
service_subclass_name	VARCHAR	イベントが発生したサービス・サブクラスの名前。 NULL 値は、サービス・サブクラスの名前が使用不可だったことを示します。

表 2. 未フォーマット・イベント表列の定義 (続き)

列名	列データ・タイプ	列の記述
service_superclass_name	VARCHAR	イベントが発生したサービス・スーパークラスの名前。NULL 値は、サービス・スーパークラスの名前が使用不可だったことを示します。
workload_name	VARCHAR	イベントが発生したワークロードの名前。 NULL 値は、ワークロードの名前が使用不可だったことを示します。

イベント・モニター・データ読み取り用の db2evmonfmt ツール

Java™ ベースの、汎用 XML パーサー・ツールである db2evmonfmt は、未フォーマット・イベント表を使用するイベント・モニターによって生成されたデータから、判読可能なフラット・テキスト出力 (テキスト・バージョン) またはフォーマット済み XML 出力を生成します。指定したパラメーターに基づいて、db2evmonfmt ツールは、イベント・モニター・データの解析方法および作成する出力のタイプを決定します。

db2evmonfmt ツールは、Java ソース・コードとして提供されます。使用する前に、以下のステップを実行して、このツールをセットアップおよびコンパイルする必要があります。

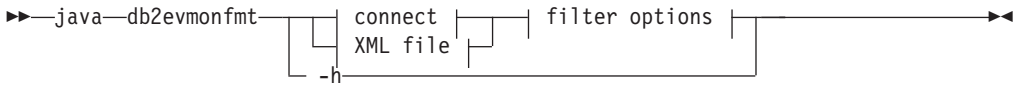
1. sqllib/samples/java/jdbc ディレクトリーでソース・コードを見つける。
2. Java ソース・ファイルに組み込まれている説明に従って、ツールをセットアップおよびコンパイルする。

ソース・コードは、ユーザーの好みに合わせて出力を変えるために変更できます。

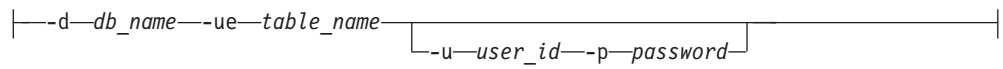
このツールは、XSLT スタイル・シートを使用して、イベント・データをフォーマット済みテキストにトランスフォームします。これらのスタイル・シートについては理解する必要はありません。ツールは、イベント・モニター・タイプに基づいて正しいスタイル・シートを自動的にロードし、イベント・データをトランスフォームします。各イベント・モニターは、sqllib/samples/xml/data ディレクトリー内でデフォルトのスタイル・シートを提供します。さらにツールは、以下のフィルター・オプションを提供します。

- イベント ID
- イベント・タイム・スタンプ
- イベント・タイプ
- ワークロード名
- サービス・クラス名
- アプリケーション名

ツールの構文

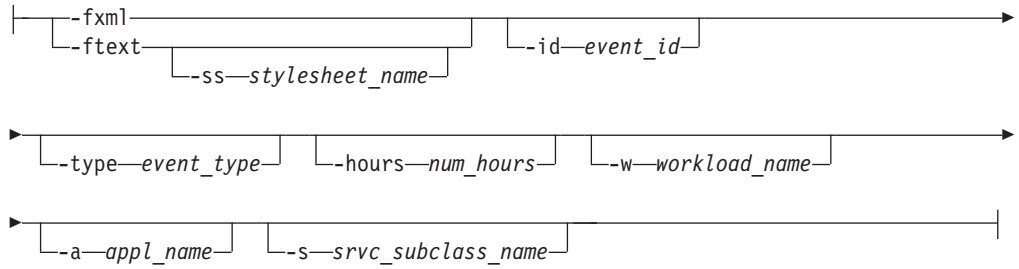


connect:



XML file:

filter options:



ツールのパラメーター

java
db2evmonfmt Java ベース・ツールを正常に実行するには、java キーワードをツール名の先頭に付ける必要があります。このツールを正常に実行するための正しい Java バージョンは、DB2 製品のインストール中に sqllib/java/jdk64 ディレクトリーからインストールされます。

-d *db_name*
接続先のデータベース名を指定します。

-ue *table_name*
未フォーマット・イベント表の名前を指定します。

-u *user_id*
ユーザー ID を指定します。

-p *password*
パスワードを指定します。

-f *xml_filename*
フォーマットする入力 XML ファイルの名前を指定します。

-fxml
フォーマット済み XML 文書を生成します (STDOUT にパイプ)。

-ftext
XML 文書をフォーマットしてテキスト文書にします (STDOUT にパイプ)。

-ss *stylesheet_name*
XML 文書のトランスフォームに使用する XSLT スタイル・シートを指定します。

-id *event_id*
指定したイベント ID に一致するすべてのイベントを表示します。

-type *event_type*
指定したイベント・タイプに一致するすべてのイベントを表示します。

-hours *num_hours*
指定した直近の数時間内に発生したすべてのイベントを表示します。

-w *workload_name*
指定したワークロードの一部であるすべてのイベントを表示します。

-a *appl_name*
指定したアプリケーションの一部であるすべてのイベントを表示します。

-s *srvc_subclass_name*
指定したサービス・サブクラスの一部であるすべてのイベントを表示します。

XSLT スタイル・シート

DB2 データベース・マネージャーは、デフォルトの XSLT スタイル・シートを提供しています (表 1 を参照)。これは sqllib/samples/java/jdbc ディレクトリー内にあります。これらのスタイル・シートは、必要とする出力を生成するために変更できます。

表 3. イベント・モニター用のデフォルトの XSLT スタイル・シート

イベント・モニター	デフォルトの XSLT スタイル・シート
ロック	DB2EvmonLocking.xsl
作業単位	DB2EvmonUOW.xsl

XML 文書をトランスフォームするための独自の XSLT スタイル・シートを作成できます。これらのスタイル・シートは、-ss stylesheet_name オプションを使用して Java ベース・ツールに渡すことができます。

例

例 1 直近の 32 時間内に発生したすべてのイベントについてのフォーマット済みテキスト出力を、データベース SAMPLE にあるパッケージ・キャッシュの未フォーマット・イベント表 PKGから得るには、以下のコマンドを発行します。

```
java db2evmonfmt -d sample -ue pkg -ftext -hours 32
```

例 2 直近の 24 時間内に発生した LOCKTIMEOUT タイプのすべてのイベントについてのフォーマット済みテキスト出力を、データベース SAMPLE にある未フォーマット・イベント表 LOCK から得るには、以下のコマンドを発行します。

```
java db2evmonfmt -d sample -ue LOCK -ftext -hours 24 –type locktimeout
```

例 3 直近の 5 時間内で、イベント・タイプ LOCKWAIT と一致するすべてのイベントを抽出するために、XML ソース・ファイル LOCK.XML からフォーマット済みテキスト出力を得るには、以下のコマンドを発行します。

```
java db2evmonfmt -f lock.xml -ftext -type lockwait -hours 5
```

例 4 データベース SAMPLE 内の未フォーマット・イベント表 UOW にあるすべてのイベントについて、作成済みの XSLT スタイル・シート SUMMARY.XSL を使用してフォーマット済みテキスト出力を得るには、以下のコマンドを発行します。

```
java db2evmonfmt -d sample -ue uow -ftext -ss summary.xsl
```

サンプルのフォーマット済みフラット・テキスト出力

ロック・イベント・モニターの XSLT スタイル・シートから生成されたフォーマット済みフラット・テキスト出力のサンプルを以下に示します。

```
------------------------------------------------
Event Entry     : 0
Event ID        : 1
Event Type      : Locktimeout
Event Timestamp : 2008-05-23-12.00.14.132329000
------------------------------------------------

Lock Details
------------
Lock Name       : 02000401000000000000000054
Lock Type       : Table
Lock Attributes : 00000000
Lock Count      : 1
Lock Hold Count : 0
Lock rrIID      : 0
Lock Status     : Waiting
Cursor Bitmap   : 00000000
Tablespace Name : USERSPACE1
Table Name      : NEWTON  .SARAH

Attributes           Requestor                      Holder
----------           ---------                      ------
Application Handle   [0-35]                         [0-16]
Application ID       *LOCAL.horton.080523160016    *LOCAL.horton.080523155938
Application Name     xaplus0001                     db2bp
Authentication ID    NEWTON                         HORTON
Requesting Agent     65                             21
Coordinating Agent   65                             21
Application Status   SQLM_CONNECTPEND               SQLM_CONNECTPEND
Lock Timeout         5000                           0
Workload Name        XAPLUS0010_WL02                SYSDEFAULTUSERWORKLOAD
Service Subclass     XAPLUS0010_SC02                SYSDEFAULTSUBCLASS
Current Request      Execute                        Execute Immediate
Lock Mode            Intent Exclusive               Exclusive
tpmon Userid
tpmon Wkstn
tpmon App
tpmon Accstring

Lock Requestor Current Activities
---------------------------------
Activity ID     : 2
Uow ID          : 1
Package ID      : 65426E4D4B584659
Package SectNo  : 3
Package Name    : NEWTON
Package Schema  : AKINTERF
```

```
Package Version :
Reopt           : always
Eff Isolation   : Cursor Stability
Eff Locktimeout : 5
Eff Degree      : 0
Nesting Level   : 0
Stmt Unicode    : No
Stmt Flag       : Dynamic
Stmt Type       : DML, Insert/Update/Delete
Stmt Text       : INSERT INTO SARAH VALUES(:H00008, :H00013, :H00014)


Lock Requestor Past Activities
------------------------------
Activity ID     : 1
Uow ID          : 1
Package ID      : 65426E4D4B584659
Package SectNo  : 2
Package Name    : NEWTON
Package Schema  : AKINTERF
Package Version :
Reopt           : always
Eff Isolation   : Cursor Stability
Eff Locktimeout : 5
Eff Degree      : 0
Nesting Level   : 0
Stmt Unicode    : No
Stmt Flag       : Dynamic
Stmt Type       : DML, Insert/Update/Delete
Stmt Text       : INSERT INTO NADIA VALUES(:H00007)


Lock Holder Current Activities
------------------------------

Lock Holder Past Activities
------------------------------
Activity ID     : 1
Uow ID          : 2
Package ID      : 41414141414E4758
Package SectNo  : 201
Package Name    : NULLID
Package Schema  : SQLC2G13
Package Version :
Reopt           : none
Eff Isolation   : Cursor Stability
Eff Locktimeout : 5
Eff Degree      : 0
Nesting Level   : 0
Stmt Unicode    : No
Stmt Flag       : Dynamic
Stmt Type       : DML, Select (blockable)
Stmt Text       : select * from newton.sarah

Activity ID     : 2
Uow ID          : 2
Package ID      : 41414141414E4758
Package SectNo  : 203
Package Name    : NULLID
Package Schema  : SQLC2G13
Package Version :
Reopt           : none
Eff Isolation   : Cursor Stability
Eff Locktimeout : 5
Eff Degree      : 0
Nesting Level   : 0
Stmt Unicode    : No
```

```
Stmt Flag       : Dynamic
Stmt Type       : DML, Lock Table
Stmt Text       : lock table newton.sarah in exclusive mode


-----------------------------------------------
Event Entry     : 1
Event ID        : 2
Event Type      : Locktimeout
Event Timestamp : 2008-05-23-12.04.42.144896000
-----------------------------------------------

...
...
...
```

使用上の注意

db2evmonfmt ユーティリティーは、Java ベース・ツールであり、正常に実行するには java キーワードを先頭に付ける必要があります。必要な Java バージョンは、DB2 製品と共に sqllib/java/jdk64 ディレクトリーからインストールされたものです。

注: さらに、EVMON_FORMAT_UE_TO_XML 表関数を使用して、未フォーマット・イベント表の BLOB 列に含まれているバイナリー・イベントを XML 文書にフォーマットすることもできます。

データベース・ロックのモニター

大規模な DB2 環境内でのロック競合状態の診断および修正は、複雑で時間がかかる作業となる場合があります。ロック・イベント・モニターおよび他の機能は、ロック・データを収集してこの作業を単純化するように設計されています。

概要

ロック・イベント・モニターは、ロック・イベントの発生時に、それらのロック・イベントに関する説明情報をキャプチャーするために使用します。キャプチャーされた情報により、ロック・イベントの原因となっているロック競合に関係した主要なアプリケーションを識別できます。ロック・リクエスター (デッドロックまたはロック・タイムアウト・エラーを受け取ったアプリケーション、または指定された期間を超えてロックを待機したアプリケーション)、および現在のロック所有者の両方についての情報がキャプチャーされます。

ロック・イベント・モニターにより収集された情報は、データベース内の未フォーマット・イベント表にバイナリー・フォーマットで書き込まれます。キャプチャーされたデータは、キャプチャー・プロセスの効率性を改善するポスト・キャプチャー・ステップで処理されます。

動的または静的 SQL のいずれかを使用して、ロック・イベント情報を収集するために、DB2 リレーショナル・モニター・インターフェース (表関数) に直接アクセスすることもできます。

デッドロックまたはロック・タイムアウトが発生しているかどうかの判別も、単純化されています。これらのイベントのいずれかが発生している場合、メッセージが管理通知ログに書き込まれます。これはアプリケーションに返される SQL0911N

(sqlcode -911) エラーの補足となります。加えて、ロック・エスカレーションの通知も管理通知ログに書き込まれます。この情報は、ロック表のサイズおよびアプリケーションが使用できる表の量の調整に役立てることができます。ロック・タイムアウト (**lock_timeouts**)、ロック待機 (**lock_waits**)、およびデッドロック (**deadlocks**) のカウンターもあり、これらを調べることもできます。

ロック・データのキャプチャーの対象にできるアクティビティーのタイプには、以下があります。

- SQL ステートメント。例えば以下のようなものがあります。
  - DML
  - DDL
  - CALL
- LOAD コマンド
- REORG コマンド
- BACKUP DATABASE コマンド
- ユーティリティー要求

ロック・イベント・モニターは、非推奨のデッドロック・イベント・モニター (CREATE EVENT MONITOR FOR DEADLOCKS ステートメントおよび DB2DETAILDEADLOCK)、および非推奨のロック・タイムアウト・レポート機能 (DB2_CAPTURE_LOCKTIMEOUT レジストリー変数) を、ロック・イベント・データ収集用の単純化されて一貫性があるインターフェースで置き換え、ロック待機に関するデータをキャプチャーする機能を追加します。

機能の概要

ロック・イベント・モニターを使用してロック・イベント・データをキャプチャーできるようにするには、以下の 2 つのステップが必要です。

1. CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して LOCK EVENT モニターを作成する必要があります。このモニターの名前と、ロック・イベント・データが書き込まれる未フォーマット・イベント表の名前を入力します。
2. 以下のいずれかの方式を使用して、ロック・イベント・データをキャプチャーするレベルを指定する必要があります。
   - 既存のワークロードを変更するか、CREATE または ALTER WORKLOAD ステートメントを使用して新規ワークロードを作成することにより、特定のワークロードを指定することができます。ワークロード・レベルで、キャプチャーするロック・イベント・データのタイプ (デッドロック、ロック・タイムアウト、またはロック待機)、およびロックに関係したアプリケーションの SQL ステートメント履歴および入力値を必要とするかどうかを指定しなければなりません。ロック待機の場合、アプリケーションがロックを待機する期間 (その後ロック待機についてデータがキャプチャーされる) を指定することも必要です。
   - 以下の適切なデータベース構成パラメーターを設定することで、データをデータベース・レベルで収集し、すべての DB2 ワークロードに影響を与えることができます。

mon_lockwait
このパラメーターは、ロック待機イベントの生成を制御します。

ベスト・プラクティスは、ロック待機データ収集をワークロード・レベルで有効にすることです。

mon_timeout
このパラメーターは、ロック・タイムアウト・イベントの生成を制御します。

ベスト・プラクティスは、ロック・タイムアウト・データ収集がアプリケーションで予期されていない場合には、それをデータベース・レベルで有効にすることです。予期されている場合には、ワークロード・レベルで有効にします。

mon_deadlock
このパラメーターは、デッドロック・イベントの生成を制御します。

ベスト・プラクティスは、デッドロック・データ収集をデータベース・レベルで有効にすることです。

mon_lw_thresh
このパラメーターは、**mon_lockwait** のイベントが生成されるまでのロック待機時間を制御します。

SQL ステートメント履歴および入力値のキャプチャーによりオーバーヘッドが増大しますが、ロック問題を適切にデバッグするには、多くの場合このレベルの詳細情報が必要です。

ロック・イベントが発生した後に、提供されている db2evmonfmt と呼ばれる Java ベース・アプリケーションを使用して、未フォーマット・イベント表内のバイナリー・データを XML またはテキスト文書にトランスフォームすることができます。さらに、未フォーマット・イベント表の BLOB 列内のバイナリー・イベント・データは、EVMON_FORMAT_UE_TO_XML 表関数を使用して XML レポート文書にフォーマットするか、または EVMON_FORMAT_UE_TO_TABLES プロシージャーを使用してリレーショナル表にフォーマットするかのいずれかが可能です。

ロック・イベントについてモニター対象とするワークロードを判別するための助けとして、管理通知ログを調べることができます。デッドロックまたはロック・タイムアウトが検出されるたびに、メッセージがそのログに書き込まれます。これらのメッセージでは、ロック・リクエスターまたはロック所有者 (複数の場合あり) が実行しているワークロード、およびロック・イベントのタイプが示されます。ロック・タイムアウト (**lock_timeouts**)、ロック待機 (**lock_waits**)、およびデッドロック (**deadlocks**) のワークロード・レベルのカウンターもあり、これらを調べることもできます。

ロック・イベントについて収集される情報

ロック・イベント・モニターにより収集されるロック・イベントの情報のいくつかには、以下があります。

- イベントの原因となったロック
- ロック・イベントの原因となったロックを保持しているアプリケーション

- ロック・イベントの原因となるロックを待機または要求していたアプリケーション
- ロック・イベント時のアプリケーションの実行内容

制限

- 未フォーマット・イベント表に書き込まれたロック・イベント・データの自動パージはありません。データは表から定期的にパージする必要があります。
- 収集したイベント・モニター・データは、未フォーマット・イベント表にのみ出力することができます。ファイル、パイプ、および表への出力はサポートされていません。
- ロック・イベント・モニターは、データベースごとに 1 つのみ作成するように提案されています。追加の各イベント・モニターは、同じデータのコピーを作成するだけです。

推奨されないロック・モニター機能

デフォルトでは、データベースごとに、推奨されなくなった詳細なデッドロック・イベント・モニター DB2DETAILDEADLOCK が作成され、データベースが活動化される時にそれが開始します。DB2DETAILDEADLOCK イベント・モニターは、無効にして除去する必要があります。そうしないと、推奨されないイベント・モニターと新規イベント・モニターの両方がデータを収集し、パフォーマンスにかなりの影響を与えることになります。

DB2DETAILDEADLOCK イベント・モニターを除去するには、以下の SQL ステートメントを発行します。

```
SET EVENT MONITOR DB2DETAILDEADLOCK state 0
DROP EVENT MONITOR DB2DETAILDEADLOCK
```

ロック・イベント・データの収集とレポートの生成

ロック・イベント・モニターを使用してロック・タイムアウト、ロック待機、およびデッドロック情報を収集することにより、ロッキングに関する問題の識別と解決に役立てることができます。ロック・イベント・データは読めない形式で未フォーマット・イベント表に収集されるので、このタスクでは、可読テキスト・レポートを後から取得する方法について説明します。

始める前に

ロッキング・イベント・モニターを作成してロック・イベント・モニター・データを収集するには、DBADM または SQLADM 権限がなければなりません。

このタスクについて

ロック・イベント・モニターは、ロッキングに関する問題の識別と解決に役立つ関連情報を収集します。ロック・イベント・モニターが収集するロック・イベントの情報には、例えば以下の情報が含まれます。

- ロック・イベントになったロック。
- ロック・イベントになったロックを要求または保持しているアプリケーション。
- ロック・イベント時のアプリケーションの実行内容。

このタスクでは、特定のワークロードに関するロック・イベント・データを収集するための手順を説明します。次のような状況下では、ロック・イベント・データを収集することをお勧めします。

- MON_GET_WORKLOAD 表関数を使用したところ、ロック待機の値がいつもより長い。
- アプリケーションが「ロック・タイムアウトのために、トランザクションがロールバックされました」という内容の -911 SQL 戻りコードを理由コード 68 と共に戻す。
- 管理通知ログにデッドロック・イベント・メッセージがある。ログ・メッセージが、2 つのアプリケーション (例えば、ワークロード FINANCE の一部であるアプリケーション A とワークロード PAYROLL の一部であるアプリケーション B) の間でロック・イベントが発生したことを示している。

制約事項

データ値を表示するには、EVMON_FORMAT_UE_* ルーチンに対する EXECUTE 特権が必要です。この特権は、SQLADM および DBADM 権限が暗黙的に保持しています。未フォーマット・イベント表に対する SELECT 特権も必要です。DATAACCESS 権限を持つユーザーと、イベント・モニターおよび関連する未フォーマット・イベント表の作成者は、デフォルトでこの特権を保持しています。

手順

今後発生する可能性のあるロック・イベントに関する詳細情報を収集するには、以下のステップを実行します。

1. 次の例に示すように、CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、lockevmon というロック・イベント・モニターを作成します。

   ```
   CREATE EVENT MONITOR lockevmon FOR LOCKING
      WRITE TO UNFORMATTED EVENT TABLE
   ```

   注: 以下は、イベント・モニターを作成する際に覚えておかなければならない重要点のリストです。

   - イベント・モニターは前もって作成できます。また、ディスク・スペースが使い尽くされることを心配する必要はありません。データベース・レベルまたはワークロード・レベルのデータ収集を活動化するまでは何も書き込まれません。
   - パーティション・データベース環境では、すべてのノードにわたるパーティション表スペースにイベント・モニターを配置するようにします。こうしないと、パーティション表スペースが存在しないパーティションでは、ロック・イベントが見落とされることになります。
   - データ取得のための表へのアクセス時に進行中の作業によるハイパフォーマンス作業に対する妨害が最小になるように、表スペースとバッファー・プールをセットアップするようにします。
2. ステートメント履歴を指定した ALTER WORKLOAD ステートメントを使用して、ワークロード・レベルのロック・イベント・データ収集を使用可能にします。データベース構成パラメーターを設定すると、データベース・レベルのロック・イベント・データ収集に影響を与え、すべてのワークロードが影響を受けます。

ロック待機イベントの場合

FINANCE アプリケーションについては 5 秒経過後に獲得されたロックのロック待機データを収集し、PAYROLL アプリケーションについては 10 秒経過後に獲得されたロックのロック待機データを収集するには、以下のステートメントを発行します。

```
ALTER WORKLOAD finance COLLECT LOCK WAIT DATA WITH HISTORY AND VALUES
   FOR LOCKS WAITING MORE THAN 5 SECONDS
ALTER WORKLOAD payroll COLLECT LOCK WAIT DATA
   FOR LOCKS WAITING MORE THAN 10 SECONDS WITH HISTORY
```

SAMPLE データベースの **mon_lockwait** データベース構成パラメーターを HIST_AND_VALUES 入力データ値を指定して設定し、**mon_lw_thresh** データベース構成パラメーターを 10 秒に設定するには、以下のコマンドを発行します。

```
db2 update db cfg for sample using mon_lockwait hist_and_values
db2 update db cfg for sample using mon_lw_thresh 10000000
```

ロック・タイムアウト・イベントの場合

FINANCE および PAYROLL アプリケーションのロック・タイムアウト・データを収集するには、以下のステートメントを発行します。

```
ALTER WORKLOAD finance COLLECT LOCK TIMEOUT DATA WITH HISTORY
ALTER WORKLOAD payroll COLLECT LOCK TIMEOUT DATA WITH HISTORY
```

SAMPLE データベースの **mon_locktimeout** データベース構成パラメーターを HIST_AND_VALUES 入力データ値を指定して設定するには、次のコマンドを発行します。

```
db2 update db cfg for sample using mon_locktimeout hist_and_values
```

デッドロック・イベントの場合

FINANCE および PAYROLL アプリケーションのデータを収集するには、以下のステートメントを発行します。

```
ALTER WORKLOAD finance COLLECT DEADLOCK DATA WITH HISTORY
ALTER WORKLOAD payroll COLLECT DEADLOCK DATA WITH HISTORY
```

SAMPLE データベースの **mon_deadlock** データベース構成パラメーターを HIST_AND_VALUES 入力データ値を指定して設定するには、次のコマンドを発行します。

```
db2 update db cfg for sample using mon_deadlock hist_and_values
```

3. 別のロック・イベント通知を受け取るために、ワークロードを再実行します。
4. データベースに接続します。
5. 以下のいずれかのアプローチで、ロッキング・イベント・レポートを取得します。
   a. XML パーサー・ツール db2evmonfmt を使用して、未フォーマット・イベント表に収集されたイベント・データに基づいた、デフォルトのスタイルシートを使用するフラット・テキスト・レポートを生成します。例えば、次のようにします。

      ```
      java db2evmonfmt -d db_name -ue table_name -ftext -u user_id -p password
      ```

   b. EVMON_FORMAT_UE_TO_XML 表関数を使用して、XML 文書を取得します。

c. EVMON_FORMAT_UE_TO_TABLES プロシージャーを使用して、データをリレーショナル表に出力します。

6. レポートを分析し、ロック・イベントに関する問題の理由を判別して、問題を解決します。
7. 以下のステートメントを実行するか、データベース構成パラメーターを再設定して、FINANCE と PAYROLL の両方のアプリケーションのロック・データ収集をオフにします。

ロック待機イベントの場合

```
ALTER WORKLOAD finance COLLECT LOCK WAIT DATA NONE
ALTER WORKLOAD payroll COLLECT LOCK WAIT DATA NONE
```

SAMPLE データベースの **mon_lockwait** データベース構成パラメーターをデフォルトの NONE 入力データ値を指定して再設定し、**mon_lw_thresh** データベース構成パラメーターを再設定してデフォルト値の 5 秒に戻すには、以下のコマンドを発行します。

```
db2 update db cfg for sample using mon_lockwait none
db2 update db cfg for sample using mon_lw_thresh 5000000
```

ロック・タイムアウト・イベントの場合

```
ALTER WORKLOAD finance COLLECT LOCK TIMEOUT DATA NONE
ALTER WORKLOAD payroll COLLECT LOCK TIMEOUT DATA NONE
```

SAMPLE データベースの **mon_locktimeout** データベース構成パラメーターをデフォルトの NONE 入力データ値を指定して再設定するには、次のコマンドを発行します。

```
db2 update db cfg for sample using mon_locktimeout none
```

デッドロック・イベントの場合

```
ALTER WORKLOAD finance COLLECT DEADLOCK DATA NONE
ALTER WORKLOAD payroll COLLECT DEADLOCK DATA NONE
```

SAMPLE データベースの **mon_deadlock** データベース構成パラメーターをデフォルトの WITHOUT_HIST 入力データ値を指定して再設定するには、次のコマンドを発行します。

```
db2 update db cfg for sample using mon_deadlock without_hist
```

次の作業

該当アプリケーションを再実行して、ロッキングに関する問題が除去されたことを確認します。

ロック・イベント・モニター用に XML に書き込まれる情報

EVMON_FORMAT_UE_TO_XML 表関数からロック・イベント・モニター用に書き込まれる情報。これは、sqllib/misc/DB2EvmonLocking.xsd ファイルにも記載されています。

db2_lock_event

ロック・タイムアウト、ロック待機、またはデッドロック・イベントを詳述するメイン・スキーマ。

表 4. db2_lock_event

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
db2_lock_event	db2lockevent	1	1	ロック・タイムアウト、ロック待機、またはデッドロック・イベントを詳述するメイン・スキーマ。

db2lockevent

このスキーマは、イベント・モニターによってキャプチャーされた各ロック・イベントの構造を記述します。

表 5. db2lockevent 属性

名前	データ・タイプ	使用	説明
id	xs:positiveInteger	必須	イベント ID を表す整数。
type	xs:string	必須	発生したロック・イベントのタイプ。ロック・イベントのタイプは、Locktimeout、Deadlock、または Lockwait になります。
timestamp	db2_timestamp_type	必須	ロック・イベントが発生した時刻を表すタイム・スタンプ。
member	member_type	必須	ロック・イベントが発生したメンバー。
release	xs:nonNegativeInteger	必須	このイベントがキャプチャーされた DB2 製品レベルを表します。

通常のイベントでは以下のエレメントが戻されます。

表 6. 通常のイベントで戻される db2lockevent エレメント

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
db2_deadlock_graph	db2dgraph	0	1	スキーマ・エレメントは DB2 Deadlock Graph を表します。このグラフは、デッドロックに関係するすべての参加者を略述します。
db2_participant	db2appinfo	1	無制限	スキーマ・エレメントは、ロック・イベントに関係するすべての参加者のアプリケーション情報を表します。

エラー状態が発生した場合、メッセージの詳細を示す以下のエレメントが戻されます。

表 7. エラー状態で戻される db2lockevent エレメント

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
db2_message	xs:string	1	1	エラー・メッセージ
db2_event_file	xs:string	1	1	イベントが書き込まれたファイルの絶対パス。

db2appdetails

このスキーマは、参加者に関する詳細を表します。

表 8. db2appdetails エレメント

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
application_handle	application_handle_type	1	1	システム全体での、アプリケーションのユニーク ID。詳しくは、モニター・エレメント agent_id を参照してください。
appl_id	appl_id_type	1	1	アプリケーションがデータベース・マネージャーでデータベースに接続すると、この ID が生成されます。詳しくは、モニター・エレメント appl_id を参照してください。
appl_name	xs:string	1	1	クライアントで実行中のアプリケーションの名前。データベースが識別できる名前です。詳しくは、モニター・エレメント appl_name を参照してください。
auth_id	authid_type	1	1	モニターされているアプリケーションを呼び出したユーザーの許可 ID。詳しくは、モニター・エレメント auth_id を参照してください。
agent_tid	xs:nonNegativeInteger	1	1	エージェントのエンジン・ディスパッチ可能単位 (EDU) のユニーク ID。詳しくは、モニター・エレメント agent_pid を参照してください。

表 8. *db2appdetails* エレメント (続き)

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
coord_agent_tid	xs:nonNegativeInteger	1	1	アプリケーションに関するコーディネーター・エージェントのエンジン・ディスパッチ可能単位 (EDU) ID。詳しくは、モニター・エレメント coord_agent_pid を参照してください。
appl_status	.	1	1	アプリケーションの現在の状況。詳しくは、モニター・エレメント appl_status を参照してください。
appl_action	.	1	1	クライアント・アプリケーションが実行中のアクション/要求。
lock_timeout_val	xs:integer	1	1	データベース構成パラメーターのロック・タイムアウト。値は秒単位です。詳しくは、モニター・エレメント lock_timeout_val を参照してください。
lock_wait_val	xs:integer	1	1	ロック・イベント時に有効なロック待機パラメーター。これは、ワークロード・レベルで指定されたデータベース構成パラメーター MON_LW_THRESH または COLLECT LOCK WAIT DATA 設定のいずれかになります。値はミリ秒単位です。
tentry_state	.	1	1	TEntry 状態。内部使用専用です。
tentry_flag1	xs:hexBinary	1	1	TEntry flags1。内部使用専用です。
tentry_flag2	xs:hexBinary	1	1	TEntry flags2。内部使用専用です。
xid	xs:hexBinary	1	1	XID - グローバル・トランザクション ID
workload_id	db_object_id_type	1	1	このアプリケーションが所属するワークロードの ID。詳しくは、モニター・エレメント workload_id を参照してください。

表 8. *db2appdetails エレメント (続き)*

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
workload_name	db_object_name_type	1	1	このアプリケーションが所属するワークロードの名前。詳しくは、モニター・エレメント workload_name を参照してください。
service_class_id	db_object_id_type	1	1	このアプリケーションが所属するサービス・サブクラスの ID。詳しくは、モニター・エレメント service_class_id を参照してください。
service_subclass_name	db_object_name_type	1	1	このアプリケーションが所属するサービス・サブクラスの名前。詳しくは、モニター・エレメント service_subclass_name を参照してください。
current_request	xs:string	1	1	現在処理中または最後に処理された操作。
lock_escalation	xs:string	1	1	ロック要求がロック・エスカレーションの一部として行われたかどうかを示します。詳しくは、モニター・エレメント lock_escalation を参照してください。可能な値は Yes または No です。
past_activities_wrapped	xs:string	1	1	アクティビティー・リストが折り返したかどうかを示します。 1 つのアプリケーションが保持する過去のアクティビティーの数のデフォルトの限度は 250 です。このデフォルトは、レジストリー変数 DB2_MAX_INACT_STMTS を使用してオーバーライドできます。この限度に対して異なる値を選択することによって、非アクティブ・ステートメントの情報のために用いられるシステム・モニター・ヒープの量を増減できます。

表 8. db2appdetails エレメント (続き)

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
client_userid	tpmon_type	1	1	トランザクション・マネージャーが生成してサーバーに提供したクライアント・ユーザー ID。詳しくは、モニター・エレメント client_userid を参照してください。
client_wrkstnname	tpmon_type	1	1	この接続で sqleseti API が発行された場合に、クライアントのシステムまたはワークステーションを示します。詳しくは、モニター・エレメント client_wrkstnname を参照してください。
client_applname	tpmon_type	1	1	この接続で sqleseti API が発行された場合に、トランザクションを実行中のサーバー・トランザクション・プログラムを示します。詳しくは、モニター・エレメント client_applname を参照してください。
client_acctng	tpmon_type	1	1	この接続で sqleseti API が発行された場合に、ロギングおよび診断の目的でターゲット・データベースに渡されたデータです。詳しくは、モニター・エレメント client_acctng を参照してください。

db2appinfo

このスキーマは、ロックを要求しているか、またはロックを保持しているアプリケーションの構造について記述します。

表 9. db2appinfo 属性

名前	データ・タイプ	使用	説明
no	xs:positiveInteger	必須	ロック・イベント内のこの参加者を一意的に識別するシーケンス番号。
type	xs:string	必須	参加者のタイプ。参加者は、Requestor または Owner のいずれかになります。

表 9. *db2appinfo 属性 (続き)*

名前	データ・タイプ	使用	説明
participant_no_holding_lk	xs:positiveInteger	オプション	ロックを保持しているアプリケーションを識別する参加者の番号。

表 10. *db2appinfo エレメント*

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
db2_object_requested	db2objectrequested	0	1	スキーマ・エレメントは、要求者が取得しようとしている DB2 ロック (所有者によって保持されている) を表します。
db2_app_details	db2appdetails	1	1	スキーマ・エレメントは、この参加者に関する詳細を表します。
db2_activity	db2activity	0	無制限	アプリケーションが現在実行しているか、あるいはすでに実行したすべての DB2 アクティビティーのリスト。

db2objectrequested

表 11. *db2objectrequested 属性*

名前	データ・タイプ	使用	説明
type	xs:string	必須	要求されているオブジェクトのタイプ。可能な値は lock または ticket です。

属性 db2objectrequested/type が lock の値である場合、以下のエレメントが戻されます。

表 12. *lock に対して戻される db2objectrequested エレメント*

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
lock_name	xs:hexBinary	1	1	内部バイナリー・ロック名。このエレメントはロックのユニーク ID を示します。詳しくは、モニター・エレメント lock_name を参照してください。

表 12. *lock* に対して戻される *db2objectrequested* エレメント (続き)

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
lock_object_type	.	1	1	アプリケーションがロックを取得するために待機しているオブジェクトのタイプ。詳しくは、モニター・エレメント lock_object_type を参照してください。
lock_specifics	xs:string	1	1	ロックに関する内部特性。通知メッセージです。
lock_attributes	xs:hexBinary	1	1	ロックの属性。詳しくは、モニター・エレメント lock_attributes を参照してください。
lock_current_mode	.	1	1	変換する前の元のロック。詳しくは、モニター・エレメント lock_current_mode を参照してください。
lock_mode_requested	.	1	1	この参加者が要求しているロック・モード。詳しくは、モニター・エレメント lock_mode_requested を参照してください。
lock_mode	.	1	1	保持されているロックのタイプ。詳しくは、モニター・エレメント lock_mode を参照してください。
lock_count	xs:integer	1	1	保持されているロックに関するロックの数。詳しくは、モニター・エレメント lock_count を参照してください。
lock_hold_count	xs:integer	1	1	ロックに置かれている保留の数。詳しくは、モニター・エレメント lock_hold_count を参照してください。
lock_rriid	xs:integer	1	1	行ロック用の IID。内部使用専用です。
lock_status	.	1	1	ロックの内部状況を示します。詳しくは、モニター・エレメント lock_status を参照してください。
lock_release_flags	xs:hexBinary	1	1	ロック保留解除フラグ。詳しくは、モニター・エレメント lock_release_flags を参照してください。

表 12. *lock* に対して戻される *db2objectrequested* エレメント *(続き)*

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
tablespace_name	.	1	1	ロックが保持されている表スペースの名前。詳しくは、モニター・エレメント tablespace_name を参照してください。
table_name	.	1	1	ロックが保持されている表の名前。詳しくは、モニター・エレメント table_name を参照してください。
table_schema	db_object_name_type	1	1	表のスキーマ。詳しくは、モニター・エレメント table_schema を参照してください。

属性 db2objectrequested/type が ticket の値である場合、以下のエレメントが戻されます。

表 13. *ticket* に対して戻される *db2objectrequested* エレメント

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
threshold_name	db_object_name_type	1	1	しきい値キューの名前。
threshold_id	db_object_id_type	1	1	しきい値キューの ID。
queued_agents	xs:nonNegativeInteger	1	1	しきい値で現在キューに入れられているエージェント数の合計。

db2dgraph

このスキーマは、デッドロック・グラフの構造と、デッドロック・サイクルに関係している参加者の構造について記述します。

表 14. *db2dgraph* 属性

名前	データ・タイプ	使用	説明
dl_conns	xs:nonNegativeInteger	必須	デッドロックに関係している参加者数の合計。詳しくは、モニター・エレメント dl_conns を参照してください。
rolled_back_participant_no	xs:nonNegativeInteger	必須	ロールバックされたアプリケーションを識別する参加者の番号。詳しくは、モニター・エレメント rolled_back_participant_no を参照してください。

表 14. db2dgraph 属性 (続き)

名前	データ・タイプ	使用	説明
type	xs:string	必須	発生したデッドロックのタイプ: local または global。グローバル・デッドロックでは、少なくとも 1 つ以上の参加者が別々のメンバーに存在します。ローカル・デッドロックでは、すべての参加者が同じメンバーに存在します。

表 15. db2dgraph エレメント

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
db2_participant	db2dstack	1	無制限	スキーマ・エレメントは、デッドロック・グラフ内の単一スタック項目を表します。

db2dstack

このスキーマは、デッドロック・グラフ内の項目の構造を記述します。この項目には、デッドロック・サイクルでロックを要求および保持している参加者に関する情報が含まれます。

表 16. db2dstack 属性

名前	データ・タイプ	使用	説明
no	xs:positiveInteger	必須	ロックを要求しているアプリケーションを識別する参加者の番号。
deadlock_member	member_type	必須	参加者がロックを要求しているメンバー。
participant_no_holding_lk	xs:positiveInteger	必須	ロックを保持しているアプリケーションを識別する参加者の番号。
application_handle	application_handle_type	必須	システム全体での、アプリケーションのユニーク ID。詳しくは、モニター・エレメント agent_id を参照してください。

db2actdetails

このスキーマは、アクティビティーに関する詳細を記述します。

表 17. *db2actdetails* エレメント

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
activity_id	xs:positiveInteger	1	1	特定の作業単位内のアプリケーションのアクティビティーを一意的に識別するカウンター。詳しくは、モニター・エレメント activity_id を参照してください。
uow_id	xs:positiveInteger	1	1	このアクティビティー・レコードが適用される作業単位 ID。詳しくは、モニター・エレメント uow_id を参照してください。
package_name	xs:string	1	1	現在実行中の SQL ステートメントが含まれているパッケージの名前。詳しくは、モニター・エレメント package_name を参照してください。
package_schema	xs:string	1	1	アプリケーションをプリコンパイルしたユーザーの許可 ID。詳しくは、モニター・エレメント package_schema を参照してください。
package_version_id	xs:string	1	1	パッケージ・バージョンは、現在実行中の SQL ステートメントを含むパッケージのバージョン ID を示します。詳しくは、モニター・エレメント package_version_id を参照してください。
consistency_token	xs:string	1	1	パッケージ整合性トークンを使用すると、現在実行中の SQL ステートメントを含むパッケージのバージョンを識別できます。詳しくは、モニター・エレメント consistency_token を参照してください。

表 17. db2actdetails エレメント (続き)

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
section_number	xs:nonNegativeInteger	1	1	現在処理中または最後に処理された SQL ステートメントのパッケージにある内部セクション番号。詳しくは、モニター・エレメント section_number を参照してください。
reopt	xs:string	1	1	このパッケージをプリコンパイルするために使用される REOPT バインド・オプション。可能な値は NONE、ONCE、および ALWAYS です。詳しくは、REOPT バインド・オプションを参照してください。
incremental_bind	xs:string	1	1	パッケージが実行時に増分でバインドされました。可能な値は Yes または No です。
effective_isolation	.	1	1	SQL ステートメントが実行されていたときに有効であった分離の値。詳しくは、モニター・エレメント effective_isolation を参照してください。
effective_query_degree	xs:nonNegativeInteger	1	1	SQL ステートメントが実行されていたときに有効であった度合いの値。詳しくは、モニター・エレメント effective_query_degree を参照してください。
stmt_unicode	xs:string	1	1	SQL ステートメントのユニコード・フラグ。可能な値は Yes または No です。
stmt_lock_timeout	xs:integer	1	1	SQL ステートメントが実行されていたときに有効であったロック・タイムアウトの値。詳しくは、モニター・エレメント stmt_lock_timeout を参照してください。

表 17. *db2actdetails エレメント (続き)*

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
stmt_type	xs:string	1	1	処理される SQL ステートメントのタイプ。可能な値は Dynamic または Static です。詳しくは、モニター・エレメント stmt_type を参照してください。
stmt_operation	xs:string	1	1	SQL ステートメントの操作タイプ。詳しくは、モニター・エレメント stmt_operation を参照してください。
stmt_query_id	xs:nonNegativeInteger	1	1	SQL ステートメントに与えられる内部照会 ID。詳しくは、モニター・エレメント stmt_query_id を参照してください。
stmt_nest_level	xs:nonNegativeInteger	1	1	このエレメントは、ステートメントが実行されたときにそのステートメントに対して有効だった、ネストまたは再帰のレベルを示します。詳しくは、モニター・エレメント stmt_nest_level を参照してください。
stmt_invocation_id	xs:nonNegativeInteger	1	1	このエレメントは、SQL ステートメントが実行されたルーチン呼び出しの ID を示します。詳しくは、モニター・エレメント stmt_invocation_id を参照してください。
stmt_source_id	xs:nonNegativeInteger	1	1	このエレメントは、実行された SQL ステートメントのソースに付けられた内部 ID を示します。詳しくは、モニター・エレメント stmt_source_id を参照してください。
stmt_pkgcache_id	xs:nonNegativeInteger	1	1	このエレメントは、動的 SQL ステートメントの内部パッケージ・キャッシュ ID を示します。詳しくは、モニター・エレメント stmt_pkgcache_id を参照してください。

表 17. db2actdetails エレメント (続き)

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
stmt_text	xs:string	1	1	SQL ステートメントのテキスト。詳しくは、モニター・エレメント stmt_text を参照してください。

db2activity

このスキーマは、指定された作業単位内のアプリケーションの単一の DB2 アクティビティーの構造について記述します。

表 18. db2activity 属性

名前	データ・タイプ	使用	説明
type	xs:string	必須	この属性はタイプまたはアクティビティーを表します。可能な値は current または past です。

表 19. db2activity エレメント

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
db2_activity_details	db2actdetails	1	1	スキーマは、このアクティビティーに関する詳細を表します。
db2_input_variable	db2inputvar	0	無制限	スキーマ・エレメントは、SQL ステートメントに関連付けられた入力変数のリストを表します。

db2inputvar

このスキーマは、単一の入力変数の構造を記述します。

表 20. db2inputvar エレメント

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
stmt_value_index	xs:positiveInteger	1	1	このエレメントは、SQL ステートメントで使用される入力パラメーター・マーカーまたはホスト変数の位置を表します。詳しくは、モニター・エレメント stmt_value_index を参照してください。

表 20. db2inputvar エレメント (続き)

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
stmt_value_isreopt	xs:string	1	1	このエレメントは、変数がステートメント最適化のやり直し中に使用されたかどうかを示します。詳しくは、モニター・エレメント stmt_value_isreopt を参照してください。
stmt_value_isnull	xs:string	1	1	このエレメントは、SQL ステートメントに関連したデータ値が NULL 値かどうかを示します。詳しくは、モニター・エレメント stmt_value_isnull を参照してください。
stmt_value_type	xs:string	1	1	このエレメントは、SQL ステートメントに関連したデータ値のタイプのストリング表記です。詳しくは、モニター・エレメント stmt_value_type を参照してください。
stmt_value_data	xs:string	1	1	このエレメントは、SQL ステートメントに関連したデータ値のストリング表記です。詳しくは、モニター・エレメント stmt_value_data を参照してください。

ロック・イベント・モニター用にリレーショナル表に書き込まれる情報

EVMON_FORMAT_UE_TO_TABLES プロシージャーを使用してロック・イベント・モニター用に XML 文書からリレーショナル表に書き込まれる情報。これは、sqllib/misc/DB2EvmonLocking.xsd ファイルにも記載されています。

ロック・イベント・モニター用に書き込まれる情報

表 21. ロック・イベント・モニターに戻される情報: 表名: LOCK_EVENT

列名	データ・タイプ	説明
XMLID	VARCHAR(1024) NOT NULL	
EVENT_ID	BIGINT NOT NULL	
EVENT_TYPE	VARCHAR(128) NOT NULL	
EVENT_TIMESTAMP	TIMESTAMP NOT NULL	
MEMBER	SMALLINT NOT NULL	member - データベース・メンバー
DL_CONNS	BIGINT	dl_conns - デッドロックに関係している接続

表 21. ロック・イベント・モニターに戻される情報: 表名: *LOCK_EVENT (続き)*

列名	データ・タイプ	説明
ROLLED_BACK_PARTICIPANT_NO	BIGINT	rolled_back_participant_no - ロールバック参加アプリケーション

表 22. ロック・イベント・モニターに戻される情報: 表名: *LOCK_PARTICIPANTS*

列名	データ・タイプ	説明
XMLID	VARCHAR(1024) NOT NULL	
PARTICIPANT_NO	BIGINT	participant_no - デッドロック内の参加者
PARTICIPANT_TYPE	VARCHAR(10)	
PARTICIPANT_NO_HOLDING_LK	BIGINT	participant_no_holding_lk - アプリケーションが必要とするオブジェクトのロックを保留する参加者
APPLICATION_HANDLE	BIGINT	application_handle - アプリケーション・ハンドル
APPL_ID	VARCHAR(128)	appl_id - アプリケーション ID
APPL_NAME	VARCHAR(128)	appl_name - アプリケーション名
AUTH_ID	VARCHAR(128)	auth_id - 許可 ID
AGENT_TID	BIGINT	agent_pid - エンジン・ディスパッチ可能単位 (EDU)
COORD_AGENT_TID	BIGINT	coord_agent_pid - コーディネーター・エージェント ID
APPL_STATUS	BIGINT	appl_status - アプリケーションの状況
LOCK_TIMEOUT_VAL	BIGINT	lock_timeout_val - ロック・タイムアウト値
LOCK_WAIT_VAL	BIGINT	
WORKLOAD_ID	BIGINT	workload_id - ワークロード ID
WORKLOAD_NAME	VARCHAR(128)	workload_name - ワークロード名
SERVICE_CLASS_ID	BIGINT	service_class_id - サービス・クラス ID
SERVICE_SUBCLASS_NAME	VARCHAR(128)	service_subclass_name - サービス・サブクラス名
CURRENT_REQUEST	VARCHAR(32)	
LOCK_ESCALATION	CHAR(3)	lock_escalation - ロック・エスカレーション
PAST_ACTIVITIES_WRAPPED	CHAR(3)	
CLIENT_USERID	VARCHAR(64)	
CLIENT_WRKSTNNAME	VARCHAR(128)	
CLIENT_APPLNAME	VARCHAR(128)	
CLIENT_ACCTNG	VARCHAR(200)	
OBJECT_REQUESTED	VARCHAR(10)	
LOCK_NAME	CHAR(26)	lock_name - ロック名

表 22. ロック・イベント・モニターに戻される情報: 表名: LOCK_PARTICIPANTS (続き)

列名	データ・タイプ	説明
LOCK_OBJECT_TYPE	BIGINT	lock_object_type - 待機中のロック対象タイプ
LOCK_ATTRIBUTES	CHAR(8)	lock_attributes - ロック属性
LOCK_CURRENT_MODE	BIGINT	lock_current_mode - 変換前の元のロック・モード
LOCK_MODE_REQUESTED	BIGINT	lock_mode_requested - 要求ロック・モード
LOCK_MODE	BIGINT	lock_mode - ロック・モード
LOCK_COUNT	BIGINT	lock_count - ロック・カウント
LOCK_HOLD_COUNT	BIGINT	lock_hold_count - ロック保留カウント
LOCK_RRIID	BIGINT	
LOCK_STATUS	VARCHAR(10)	lock_status - ロック状況
LOCK_RELEASE_FLAGS	CHAR(8)	lock_release_flags - ロック保留解除フラグ
TABLE_FILE_ID	BIGIN	table_file_id - 表ファイル ID
TABLE_NAME	VARCHAR(128)	table_name - 表名
TABLE_SCHEMA	VARCHAR(128)	table_schema - 表スキーマ名
TABLESPACE_NAME	VARCHAR(128)	tablespace_name - 表スペース名
THRESHOLD_ID	BIGINT	
THRESHOLD_NAME	VARCHAR(128)	threshold_name - しきい値名

表 23. ロック・イベント・モニターに戻される情報: 表名: LOCK_PARTICIPANT_ACTIVITIES

列名	データ・タイプ	説明
XMLID	VARCHAR(1024) NOT NULL	
PARTICIPANT_NO	BIGINT	participant_no - デッドロック内の参加者
ACTIVITY_ID	BIGINT	activity_id - アクティビティー ID
ACTIVITY_TYPE	VARCHAR(10)	activity_type - アクティビティー・タイプ
UOW_ID	BIGINT	uow_id - 作業単位 ID
PACKAGE_NAME	VARCHAR(128)	package_name - パッケージ名
PACKAGE_SCHEMA	VARCHAR(128)	package_schema - パッケージ・スキーマ
PACKAGE_VERSION_ID	VARCHAR(64)	package_version_id - パッケージ・バージョン
CONSISTENCY_TOKEN	VARCHAR(8)	consistency_token - パッケージ整合性トークン
SECTION_NUMBER	BIGINT	section_number - セクション番号
REOPT	VARCHAR(10)	
INCREMENTAL_BIND	CHAR(3)	
EFFECTIVE_ISOLATION	BIGINT	effective_isolation - 有効な分離

表 23. ロック・イベント・モニターに戻される情報: 表名: LOCK_PARTICIPANT_ACTIVITIES (続き)

列名	データ・タイプ	説明
EFFECTIVE_QUERY_DEGREE	BIGINT	effective_query_degree - 有効な照会の度合い
STMT_LOCK_TIMEOUT	BIGINT	stmt_lock_timeout - ステートメント・ロック・タイムアウト
STMT_TYPE	VARCHAR(10)	stmt_type - ステートメント・タイプ
STMT_QUERY_ID	BIGINT	stmt_query_id - ステートメント照会 ID
STMT_NEST_LEVEL	SMALLINT	stmt_nest_level - ステートメント・ネスト・レベル
STMT_INVOCATION_ID	BIGINT	stmt_invocation_id - ステートメント呼び出し ID
STMT_SOURCE_ID	BIGINT	stmt_source_id - ステートメント・ソース ID
STMT_PKGCACHE_ID	BIGINT	stmt_pkgcache_id - ステートメント・パッケージ・キャッシュ ID
STMT_TEXT	CLOB(2097152)	stmt_text - SQL ステートメント・テキスト

表 24. ロック・イベント・モニターに戻される情報: 表名: LOCK_ACTIVITY_VALUES

列名	データ・タイプ	説明
XMLID	VARCHAR(1024) NOT NULL	
PARTICIPANT_NO	BIGINT	participant_no - デッドロック内の参加者
ACTIVITY_ID	BIGINT	activity_id - アクティビティー ID
UOW_ID	BIGINT	uow_id - 作業単位 ID
STMT_VALUE_INDEX	BIGINT	stmt_value_index - 値索引
STMT_VALUE_ISREOPT	CHAR(3)	stmt_value_isreopt - ステートメント最適化のやり直しに使用される変数
STMT_VALUE_ISNULL	CHAR(3)	stmt_value_isnull - NULL 値の値
STMT_VALUE_TYPE	CHAR(16)	stmt_value_type - 値タイプ
STMT_VALUE_DATA	CLOB (32K)	stmt_value_data - 値データ

作業単位イベントのモニター

作業単位 (UoW) イベント・モニターは、作業単位が完了した場合はいつでも、つまりコミットまたはロールバックがあるときはいつでも、イベントを記録します。個々の UoW に関する履歴情報は、チャージバックの目的 (CPU の使用に応じて課金される)、および応答時間サービス・レベル目標の順守のモニターに役立ちます。

UoW イベント・モニターは、要求メトリックを使用してシステム・パースペクティブ・モニターを実行する 1 つの方法です。作業単位イベント・モニターに最も関連のある代替手段または補足手段は、統計イベント・モニター、または MON_GET_UNIT_OF_WORK および MON_GET_UNIT_OF_WORK_DETAILS 表関数のいずれかです。

UoW イベント・モニターを作成し、ロック・イベント・モニター・データを収集するには、DBADM または SQLADM 権限が必要です。

UoW イベント・モニターの作成およびデータ収集の構成

UoW イベント・モニターを作成する前に、イベント・モニターの未フォーマット・イベント表を保管することを計画している表スペースを特定します。推奨されている構成は、イベント・モニターに関連付けられている未フォーマット・イベント表を保管するための専用の表スペースを構成することです。イベント・データが、未フォーマット・イベント表のインライン BLOB 列内に確実に入れられるように、表スペース内の作業単位イベント・モニターは、少なくとも 8K のページ・サイズを指定して作成します。 BLOB 列がインラインでなければ、未フォーマット・イベント表へのイベントの書き込みおよび読み取りのパフォーマンスは効率的でない場合があります。

データベース・マネージャーは、未フォーマット・イベント表内の event_data BLOB 列をインライン化しようとしますが、これは必ずしも可能ではありません。未フォーマット・イベント表内の行がインライン化されているかどうかを確認するには、ADMIN_IS_INLINED 関数を使用します。行がインライン化されていない場合は、行が必要とするスペースを判別するために、ADMIN_EST_INLINE_LENGTH 関数を使用します。

イベント・モニターを作成するときの他の選択肢として、任意の既存の表スペースを指定するか、またはどれも指定せずにデフォルトで選択されたものを使用することができます。

デフォルトおよびベスト・プラクティスを使用して UoW イベント・モニターをセットアップするには、以下のステップを実行します。

1. CREATE EVENT MONITOR ステートメントを発行してイベント・モニターを作成します。以下の例では、可能な場合はデフォルトを使用し、未フォーマット・イベント表を既存の表スペースに保管することを指定します。

   ```
   CREATE EVENT MONITOR MY_UOW_EVMON
     FOR UNIT OF WORK
      WRITE TO UNFORMATTED EVENT TABLE (IN MY_EVMON_TABLESPACE)
   ```

2. 収集するデータを構成します。以下のステートメントは、ごく簡単な方法を示しています。

   ```
   db2 update db cfg for dbname using mon_uow_data base
   ```

データ収集の構成

データ収集を構成するには、イベントのキャプチャー対象とするシステム・ワークロードのサブセットと、各イベントについて収集する詳細の程度も指定する必要があります。デフォルトでは、UoW データは収集されません。デフォルト設定は、以下のいずれかの設定を使用して変更できます。

- **mon_uow_data** データベース構成パラメーター
- CREATE/ALTER WORKLOAD ステートメントの COLLECT UNIT OF WORK DATA 節

データ収集では、以下のレベルが選択可能です。

None UoW データは収集されません

Base UoW データは収集されます

mon_uow_data データベース構成パラメーターまたは CREATE/ALTER WORKLOAD ステートメントの COLLECT UNIT OF WORK DATA 節のいずれかが BASE に設定される場合、これはワークロードに有効な設定です。

選択したワークロードについてのみデータ収集を有効にする場合、**mon_uow_data** データベース構成パラメーターを NONE に設定し、対象のワークロードのレベルを BASE に設定します。

要求メトリックは、UOW イベント・モニターを使用して収集できる情報のいずれかのタイプです。 UoW イベント・モニターは、要求メトリック収集の設定により影響を受けるいずれかのインターフェースです。デフォルトでは、要求メトリックは収集され、適切な表関数およびイベント・モニター (UoW イベント・モニターを含む) で報告されます。デフォルト設定は、以下のいずれかの設定を使用して変更できます。

- **mon_req_metrics** データベース構成パラメーター
- サービス・スーパークラスの CREATE/ALTER SERVICE CLASS ステートメントの COLLECT REQUEST METRICS 節。

これらの設定を変更すると、要求メトリックを報告するすべての表関数またはイベント・モニターに影響を与えます。

UoW イベント・モニターによりキャプチャーされたイベント・データへのアクセス

このタイプのイベント・モニターは、バイナリー・フォーマットのデータを、未フォーマット・イベント表に書き込みます。このデータは、以下の表関数を使用してアクセスできます。

- EVMON_FORMAT_UE_TO_XML - 未フォーマット・イベント表から XML 文書にデータを抽出します。
- EVMON_FORMAT_UE_TO_TABLES - 未フォーマット・イベント表から一連のリレーショナル表にデータを抽出します。

これらの表関数を使用して、SELECT ステートメントを使用して抽出するデータを指定します。 SELECT ステートメントにより提供される選択、順序付け、その他の面についてはすべて制御できます。

db2evmonfmt コマンドを使用して、以下のタスクを実行することもできます。

- イベント ID、イベント・タイプ、時間枠、アプリケーション、ワークロード、またはサービス・クラスの各属性に基づいて、関心対象のイベントを選択します。
- テキスト・レポートまたはフォーマット済み XML 文書のどちらの形式で出力を受け取るかを選択します。
- db2evmonfmt コマンドに提供されているものを使用する代わりに、独自の XSLT スタイル・シートを作成して、出力形式を制御します。

例えば、以下のコマンドにより、次のような UoW レポートが提供されます。

1. データベース SAMPLE で過去 24 時間に発生した UoW イベントを選択します。これらのイベント・レコードは、SAMPLE_UoW_events という未フォーマット・イベント表から取得します。
2. DB2EvmonUOW.xsl スタイル・シートを使用してフォーマット済みのテキスト出力を提供します。

```
java db2evmonfmt -d SAMPLE -ue SAMPLE_UOW_EVENTS -ftext -ss DB2EvmonUOW.xsl -hours 24
```

作業単位イベント・データの収集とレポートの生成

作業単位イベント・モニターを使用することにより、チャージバックの目的で使用できるトランザクションに関するデータを収集できます。トランザクション・イベント・データは読めない形式で未フォーマット・イベント表に収集されるので、このタスクでは、可読テキスト・レポートを後から取得する方法について説明します。

始める前に

作業単位イベント・モニター・データを収集するには、SYSADM または SYSCTRL 権限がなければなりません。

このタスクについて

作業単位イベント・モニターは、チャージバックの目的で使用できるアプリケーション・トランザクションと対応 CPU 使用量を示す関連情報を収集します。作業単位イベント・モニターが収集するトランザクション・イベントの情報には、例えば以下の情報が含まれます。

- 合計 CPU 使用時間 (TOTAL_CPU_TIME)
- アプリケーション・ハンドル (APPLICATION_HANDLE)

このタスクでは、特定のワークロードに関する作業単位イベント・データを収集するための手順を説明します。

制約事項

SYSADM または SYSCTRL 権限がない場合、入力データ値は表示できません。

手順

作業単位イベントに関する詳細情報を収集するには、以下のステップを実行します。

1. 次の例に示すように、CREATE EVENT MONITOR FOR UNIT OF WORK ステートメントを使用して、uowevmon という作業単位イベント・モニターを作成します。

   ```
   CREATE EVENT MONITOR uowevmon FOR UNIT OF WORK
      WRITE TO UNFORMATTED EVENT TABLE
   ```

2. ステートメント履歴を指定した ALTER WORKLOAD ステートメントを使用して、ワークロード・レベルの作業単位イベント・データ収集を使用可能にします。 FINANCE および PAYROLL アプリケーションの作業単位データを収集するには、以下のステートメントを発行します。

   ```
   ALTER WORKLOAD finance COLLECT UNIT OF WORK DATA WITH HISTORY
   ALTER WORKLOAD payroll COLLECT UNIT OF WORK DATA WITH HISTORY
   ```

3. 作業単位トランザクション・イベントを収集するために、ワークロードを再実行します。
4. データベースに接続します。
5. 次のアプローチで、作業単位イベント・レポートを取得します。
   a. XML パーサー・ツール db2evmonfmt を使用して、未フォーマット・イベント表に収集されたイベント・データに基づいたフラット・テキスト・レポートを生成します。例えば、次のようにします。

      ```
      java db2evmonfmt -d db_name -ue table_name -ftext -u user_id -p password
      ```
6. レポートを分析し、適切な課金を請求できるようにアプリケーションが使用している CPU 時間を判別します。
7. FINANCE と PAYROLL の両方のアプリケーションの作業単位データ収集をオフにする場合は、以下のステートメントを実行します。

   ```
   ALTER WORKLOAD finance COLLECT UNIT OF WORK DATA NONE
   ALTER WORKLOAD payroll COLLECT UNIT OF WORK DATA NONE
   ```

作業単位イベント・モニター用に XML に書き込まれる情報

EVMON_FORMAT_UE_TO_XML 表関数から作業単位イベント・モニター用に書き込まれる情報。これは、sqllib/misc/DB2EvmonUOW.xsd ファイルにも記載されています。

作業単位イベント・モニター用に書き込まれる情報

db2_uow_event: 作業単位イベントについて記述するメイン・スキーマ

表 25. エレメント

名前	データ・タイプ	説明
db2_uow_event	db2uowevent	作業単位イベントについて記述するメイン・スキーマ

db2uowevent: 作業単位イベントについて記述するメイン・スキーマ

表 26. 属性

名前	データ・タイプ	説明
id	xs:positiveInteger	イベント ID を表す整数
type	xs:string	発生した作業単位イベントのタイプ。イベントのタイプは UOW になります。
timestamp	db2_timestamp_type	UOW イベントが発生した時刻を表すタイム・スタンプ
member	member_type	UOW イベントが発生したメンバー
release	xs:nonNegativeInteger	このイベントがキャプチャーされた DB2 製品レベルを表す

表 27. エレメント

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
completion_status	xs:string	1	1	作業単位の完了状況。可能な値は、UNKNOWN、COMMIT、ROLLBACK、GLOBAL_COMMIT、GLOBAL ROLLBACK、XA_END、XA_PREPARE です
start_time	xs:dateTime	1	1	作業単位の開始時刻。詳しくは、モニター・エレメント uow_start_time を参照してください
stop_time	xs:dateTime	1	1	作業単位の停止時刻。詳しくは、モニター・エレメント uow_stop_time を参照してください
connection_time	xs:dateTime	1	1	アプリケーションがデータベース・メンバーに接続した時刻。詳しくは、モニター・エレメント conn_time を参照してください
application_name	xs:string	1	1	クライアントで実行中のアプリケーションの名前。データベースが識別できる名前です。詳しくは、モニター・エレメント appl_name を参照してください
application_handle	application_handle_type	1	1	システム全体での、アプリケーションのユニーク ID。詳しくは、モニター・エレメント agent_id を参照してください
application_id	appl_id_type	1	1	アプリケーションがデータベース・マネージャーでデータベースに接続すると、この ID が生成されます。詳しくは、モニター・エレメント appl_id を参照してください
uow_id	incrementing_id_type	1	1	このアクティビティー・レコードが適用される作業単位 ID。詳しくは、モニター・エレメント uow_id を参照してください

表 27. エレメント (続き)

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
workload_occurrence_id	incrementing_id_type	1	1	このアクティビティー・レコードが適用されるワークロード・オカレンス ID。詳しくは、モニター・エレメント workload_occurrence_id を参照してください
coord_member	member_type	1	1	この作業単位のコーディネート・メンバー。詳しくは、モニター・エレメント coord_partitition_num を参照してください
member_activation_time	xs:dateTime	1	1	このデータベース・メンバーがアクティブにされた時刻。詳しくは、モニター・エレメント db_conn_time を参照してください
workload_name	db_object_name_type	1	1	作業単位が完了したワークロードの名前。詳しくは、モニター・エレメント workload_name を参照してください
workload_id	db_object_id_type	1	1	作業単位が完了したワークロードのワークロード ID。詳しくは、モニター・エレメント workload_id を参照してください
service_superclass_name	db_object_name_type	0	1	作業単位が完了したサービス・スーパークラスの名前。詳しくは、モニター・エレメント service_superclass_name を参照してください
service_subclass_name	db_object_name_type	0	1	作業単位が完了したサービス・サブクラスの名前。詳しくは、モニター・エレメント service_subclass_name を参照してください
service_class_id	db_object_id_type	0	1	作業単位が完了したサービス・クラスのサービス・クラス ID。詳しくは、モニター・エレメント service_class_id を参照してください

表 27. エレメント (続き)

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
session_authid	authid_type	0	1	モニターされているアプリケーションを呼び出したユーザーのセッション許可 ID。詳しくは、モニター・エレメント auth_id を参照してください
system_authid	authid_type	1	1	モニターされているアプリケーションを呼び出したユーザーのシステム許可 ID。詳しくは、モニター・エレメント auth_id を参照してください
client_pid	xs:nonNegativeInteger	1	1	クライアントによってレポートされるプロセス ID。詳しくは、モニター・エレメント client_pid を参照してください
client_product_id	xs:string	1	1	クライアントの製品 ID。詳しくは、モニター・エレメント client_prdid を参照してください
client_platform	xs:nonNegativeInteger	1	1	クライアントのプラットフォーム。詳しくは、モニター・エレメント client_platform を参照してください
client_protocol	xs:string	0	1	クライアントの製品 ID。詳しくは、モニター・エレメント client_protocol を参照してください
client_userid	tpmon_type	0	1	トランザクション・マネージャーが生成してサーバーに提供したクライアント・ユーザー ID。詳しくは、モニター・エレメント client_userid を参照してください
client_wrkstnname	tpmon_type	0	1	この接続で sqleseti API が発行された場合に、クライアントのシステムまたはワークステーションを示します。詳しくは、モニター・エレメント client_wrkstnname を参照してください

表 27. エレメント (続き)

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
client_applname	tpmon_type	0	1	この接続で sqleseti API が発行された場合に、トランザクションを実行中のサーバー・トランザクション・プログラムを示します。詳しくは、モニター・エレメント client_applname を参照してください
client_acctng	tpmon_type	0	1	この接続で sqleseti API が発行された場合に、ロギングおよび診断の目的でターゲット・データベースに渡されたデータです。詳しくは、モニター・エレメント client_acctng を参照してください
local_transaction_id	xs:hexBinary	1	1	作業単位のローカル・トランザクション ID
global_transaction_id	xs:hexBinary	1	1	作業単位のグローバル・トランザクション ID
system_metrics	system_level_metrics	1	1	作業単位のメトリック。これは、すべてのシステム・レベル・メトリックが含まれる XML エレメントです。

作業単位イベント・モニター用にリレーショナル表に書き込まれる情報

EVMON_FORMAT_UE_TO_TABLES プロシージャーを使用して作業単位イベント・モニター用に XML 文書からリレーショナル表に書き込まれる情報。これは、`sqllib/misc/DB2EvmonUOW.xsd` ファイルにも記載されています。

UOW イベント・モニター用に書き込まれる情報

表 28. UOW イベント・モニターに戻される情報: 表名: UOW_EVENT

列名	データ・タイプ	説明
EVENT_ID	INTEGER NOT NULL	
TYPE	VARCHAR(128) NOT NULL	
EVENT_TIMESTAMP	TIMESTAMP NOT NULL	
MEMBER	SMALLINT	member - データベース・メンバー
COORD_MEMBER	SMALLINT	coord_member - コーディネーター・メンバー
COMPLETION_STATUS	VARCHAR(128)	
START_TIME	TIMESTAMP	start_time - イベント開始時刻

表 28. UOW イベント・モニターに戻される情報: 表名: UOW_EVENT (続き)

列名	データ・タイプ	説明
STOP_TIME	TIMESTAMP	stop_time - イベント停止時刻
WORKLOAD_NAME	VARCHAR(128)	workload_name - ワークロード名
WORKLOAD_ID	INTEGER	workload_id - ワークロード ID
SERVICE_SUPERCLASS_NAME	VARCHAR(128)	service_superclass_name - サービス・スーパークラス名
SERVICE_SUBCLASS_NAME	VARCHAR(128)	service_subclass_name - サービス・サブクラス名
SERVICE_CLASS_ID	INTEGER	service_class_id - サービス・クラス ID
UOW_ID	INTEGER	uow_id - 作業単位 ID
WORKLOAD_OCCURRENCE_ID	INTEGER	workload_occurrence_id - ワークロード・オカレンス ID
CONNECTION_TIME	TIMESTAMP	
MEMBER_ACTIVATION_TIME	TIMESTAMP	
APPLICATION_ID	VARCHAR(128)	
APPLICATION_HANDLE	BIGINT	application_handle - アプリケーション・ハンドル
APPLICATION_NAME	VARCHAR(128)	
SYSTEM_AUTHID	VARCHAR(128)	
SESSION_AUTHID	VARCHAR(128)	
CLIENT_PLATFORM	INTEGER	client_platform - クライアント・オペレーティング・プラットフォーム
CLIENT_PID	INTEGER	client_pid - クライアント・プロセス ID
CLIENT_PRODUCT_ID	VARCHAR(128)	
CLIENT_PROTOCOL	VARCHAR(10)	client_protocol - クライアント通信プロトコル
CLIENT_WRKSTNNAME	VARCHAR(128)	
CLIENT_ACCTNG	VARCHAR(200)	
CLIENT_USERID	VARCHAR(64)	
CLIENT_APPLNAME	VARCHAR(128)	
LOCAL_TRANSACTION_ID	VARCHAR(16)	
GLOBAL_TRANSACTION_ID	VARCHAR(40)	
METRICS	BLOB(100K)	すべてのシステム・レベル・メトリックが含まれる XML 文書。

統計イベント・モニターを使用したシステム・モニター・エレメントのキャプチャー

統計イベント・モニターには、event_scstats および event_wlstats 論理データ・グループの details_xml モニター・エレメントが含まれます。このモニター・エレメントを使用して、システムに関する情報をキャプチャーします。

モニター・エレメント details_xml は、MON_GET_SERVICE_SUBCLASS_DETAILS および MON_GET_WORKLOAD_DETAILS 表関数により報告されるすべてのシステム・モニター・エレメントを含む XML 文書です。システム・モニター・エレメントは、MON_GET_SERVICE_SUBCLASS_DETAILS および MON_GET_WORKLOAD_DETAILS 表関数の DETAILS 列で報告される詳細文書のサブセットです。

要求モニター・エレメントは、サービス・スーパークラスの COLLECT REQUEST METRICS 節、およびデータベース・レベルの mon_req_metrics データベース構成パラメーターにより制御されます。モニター・エレメントが収集されるのは、親のサービス・スーパークラスで要求モニター・エレメント収集が有効になっているサービス・サブクラス内のエージェントによって要求が処理される場合、または要求モニター・エレメント収集がデータベース全体に対して有効になっている場合のみです。データベース・レベルで、サービス・スーパークラスについて、要求モニター・エレメントが無効になっている場合、DETAILS_XML 文書で報告されるメトリックは増分を停止します (または要求メトリックがデータベース活動化の時点で無効になっていた場合は 0 のままです)。

DETAILS_XML 列で返される XML 文書のスキーマは、ファイル sqllib/misc/DB2MonCommon.xsd から入手可能です。最上位エレメントは、system_metrics です。

system_metrics モニター・エレメントの XML スキーマ

system_metrics モニター・エレメントには、MON_GET_SERVICE_SUBCLASS_DETAILS および MON_GET_WORKLOAD_DETAILS 表関数によってレポートされるすべてのシステム・メトリックが含まれます。

system_metrics

システム・レベル・メトリック。

表 29. system_metrics

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
system_metrics	system_level_metrics	1	1	システム・レベル・メトリック。

system_level_metrics

このタイプは、システム・レベルの一部であるメトリックを定義します。

表 30. system_level_metrics 属性

名前	データ・タイプ	使用	説明
release	xs:nonNegativeInteger	必須	このイベントがキャプチャーされた DB2 製品レベルを表します。

表 31. system_level_metrics エレメント

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
WLM_QUEUE_TIME_TOTAL	metric_value_type	1	1	wlm_queue_time_total - ワークロード・マネージャー合計キュー時間
WLM_QUEUE_ASSIGNMENTS_TOTAL	metric_value_type	1	1	wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て
FCM_TQ_RECV_WAIT_TIME	metric_value_type	1	1	fcm_tq_recv_wait_time - FCM 表キュー受信待機時間
FCM_MESSAGE_RECV_WAIT_TIME	metric_value_type	1	1	fcm_message_recv_wait_time - FCM メッセージの受信待機時間
FCM_TQ_SEND_WAIT_TIME	metric_value_type	1	1	fcm_tq_send_wait_time - FCM 表キュー送信待機時間
FCM_MESSAGE_SEND_WAIT_TIME	metric_value_type	1	1	fcm_message_send_wait_time - FCM メッセージの送信待機時間
AGENT_WAIT_TIME	metric_value_type	1	1	agent_wait_time - エージェント待機時間
AGENT_WAITS_TOTAL	metric_value_type	1	1	agent_waits_total - エージェント待機の合計
LOCK_WAIT_TIME	metric_value_type	1	1	lock_wait_time - ロック待機中の時間
LOCK_WAITS	metric_value_type	1	1	lock_waits - ロック待機数
DIRECT_READ_TIME	metric_value_type	1	1	direct_read_time - 直接読み取り時間
DIRECT_READ_REQS	metric_value_type	1	1	direct_read_reqs - 直接読み取り要求
DIRECT_WRITE_TIME	metric_value_type	1	1	direct_write_time - 直接書き込み時間
DIRECT_WRITE_REQS	metric_value_type	1	1	direct_write_reqs - 直接書き込み要求
LOG_BUFFER_WAIT_TIME	metric_value_type	1	1	log_buffer_wait_time - ログ・バッファー待機時間
NUM_LOG_BUFFER_FULL	metric_value_type	1	1	num_log_buffer_full - フル・ログ・バッファーの回数
LOG_DISK_WAIT_TIME	metric_value_type	1	1	log_disk_wait_time - ログ・ディスク待機時間
LOG_DISK_WAITS_TOTAL	metric_value_type	1	1	log_disk_waits_total - ログ・ディスク待機の合計
TCPIP_RECV_WAIT_TIME	metric_value_type	1	1	tcpip_recv_wait_time - TCP/IP 受信待機時間
TCPIP_RECVS_TOTAL	metric_value_type	1	1	tcpip_recvs_total - TCP/IP 受信の合計
CLIENT_IDLE_WAIT_TIME	metric_value_type	1	1	client_idle_wait_time - クライアントのアイドル待機時間
IPC_RECV_WAIT_TIME	metric_value_type	1	1	ipc_recv_wait_time - プロセス間通信受信待機時間
IPC_RECVS_TOTAL	metric_value_type	1	1	ipc_recvs_total - プロセス間通信受信の合計
IPC_SEND_WAIT_TIME	metric_value_type	1	1	ipc_send_wait_time - プロセス間通信送信待機時間

表 31. system_level_metrics エレメント (続き)

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
IPC_SENDS_TOTAL	metric_value_type	1	1	ipc_sends_total - プロセス間通信送信の合計
TCPIP_SEND_WAIT_TIME	metric_value_type	1	1	tcpip_send_wait_time - TCP/IP 送信待機時間
TCPIP_SENDS_TOTAL	metric_value_type	1	1	tcpip_sends_total - TCP/IP 送信の合計
POOL_WRITE_TIME	metric_value_type	1	1	pool_write_time - バッファー・プール物理書き込み時間の合計
POOL_READ_TIME	metric_value_type	1	1	pool_read_time - バッファー・プール物理読み取り時間の合計
AUDIT_FILE_WRITE_ WAIT_TIME	metric_value_type	1	1	audit_file_write_wait_time - 監査ファイル書き込み待機時間
AUDIT_FILE_WRITES_TOTAL	metric_value_type	1	1	audit_file_writes_total - 書き込まれた監査ファイルの合計
AUDIT_SUBSYSTEM_ WAIT_TIME	metric_value_type	1	1	audit_subsystem_wait_time - 監査サブシステム待機時間
AUDIT_SUBSYSTEM_ WAITS_TOTAL	metric_value_type	1	1	audit_subsystem_waits_total - 監査サブシステム待機の合計
DIAGLOG_WRITE_WAIT_TIME	metric_value_type	1	1	diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間
DIAGLOG_WRITES_TOTAL	metric_value_type	1	1	diaglog_writes_total - 診断ログ・ファイル書き込みの合計
FCM_SEND_WAIT_TIME	metric_value_type	1	1	fcm_send_wait_time - FCM 送信待機時間
FCM_RECV_WAIT_TIME	metric_value_type	1	1	fcm_recv_wait_time - FCM 受信待機時間
TOTAL_WAIT_TIME	metric_value_type	1	1	total_wait_time - 合計待機時間
TOTAL_RQST_TIME	metric_value_type	1	1	total_rqst_time - 合計要求時間
RQSTS_COMPLETED_TOTAL	metric_value_type	1	1	rqsts_completed_total - 完了した要求の合計
TOTAL_APP_RQST_TIME	metric_value_type	1	1	total_app_rqst_time - 合計アプリケーション要求時間
APP_RQSTS_COMPLETED_ TOTAL	metric_value_type	1	1	app_rqsts_completed_total - 完了したアプリケーション要求の合計
TOTAL_SECTION_SORT_ PROC_TIME	metric_value_type	1	1	total_section_sort_proc_time - セクション・ソート処理時間の合計
TOTAL_SECTION_SORT_TIME	metric_value_type	1	1	total_section_sort_time - セクション・ソート時間の合計
TOTAL_SECTION_SORTS	metric_value_type	1	1	total_section_sorts - セクション・ソートの合計
ROWS_READ	metric_value_type	1	1	rows_read - 読み取り行数
ROWS_MODIFIED	metric_value_type	1	1	rows_modified - 変更行数
POOL_DATA_L_READS	metric_value_type	1	1	pool_data_l_reads - バッファー・プール・データの論理読み取り
POOL_INDEX_L_READS	metric_value_type	1	1	pool_index_l_reads - バッファー・プール索引の論理読み取り
POOL_TEMP_DATA_L_READS	metric_value_type	1	1	pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り

表 31. system_level_metrics エレメント (続き)

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
POOL_TEMP_INDEX_L_READS	metric_value_type	1	1	pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り
POOL_XDA_L_READS	metric_value_type	1	1	pool_xda_l_reads - バッファー・プール XDA データの論理読み取り
POOL_TEMP_XDA_L_READS	metric_value_type	1	1	pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り
TOTAL_CPU_TIME	metric_value_type	1	1	total_cpu_time - 合計 CPU 時間
ACT_COMPLETED_TOTAL	metric_value_type	1	1	act_completed_total - 完了したアクティビティーの合計
POOL_DATA_P_READS	metric_value_type	1	1	pool_data_p_reads - バッファー・プール・データの物理読み取り
POOL_TEMP_DATA_P_READS	metric_value_type	1	1	pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り
POOL_XDA_P_READS	metric_value_type	1	1	pool_xda_p_reads - バッファー・プール XDA データの物理読み取り
POOL_TEMP_XDA_P_READS	metric_value_type	1	1	pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り
POOL_INDEX_P_READS	metric_value_type	1	1	pool_index_p_reads - バッファー・プール索引の物理読み取り
POOL_TEMP_INDEX_P_READS	metric_value_type	1	1	pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り
POOL_DATA_WRITES	metric_value_type	1	1	pool_data_writes - バッファー・プールへのデータの書き込み
POOL_XDA_WRITES	metric_value_type	1	1	pool_xda_writes - バッファー・プール XDA データの書き込み
POOL_INDEX_WRITES	metric_value_type	1	1	pool_index_writes - バッファー・プール索引の書き込み
DIRECT_READS	metric_value_type	1	1	direct_reads - データベースからの直接読み取り
DIRECT_WRITES	metric_value_type	1	1	direct_writes - データベースへの直接書き込み
ROWS_RETURNED	metric_value_type	1	1	rows_returned - 戻り行数
DEADLOCKS	metric_value_type	1	1	deadlocks - デッドロック検出数
LOCK_TIMEOUTS	metric_value_type	1	1	lock_timeouts - ロック・タイムアウト数
LOCK_ESCALS	metric_value_type	1	1	lock_escals - ロック・エスカレーション数
FCM_SENDS_TOTAL	metric_value_type	1	1	fcm_sends_total - FCM 送信の合計
FCM_RECVS_TOTAL	metric_value_type	1	1	fcm_recvs_total - FCM 受信の合計
FCM_SEND_VOLUME	metric_value_type	1	1	fcm_send_volume - FCM 送信ボリューム

表 31. system_level_metrics エレメント (続き)

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
FCM_RECV_VOLUME	metric_value_type	1	1	fcm_recv_volume - FCM 受信ボリューム
FCM_MESSAGE_SENDS_TOTAL	metric_value_type	1	1	fcm_message_sends_total - FCM メッセージ送信の合計
FCM_MESSAGE_RECVS_TOTAL	metric_value_type	1	1	fcm_message_recvs_total - FCM メッセージ受信の合計
FCM_MESSAGE_SEND_VOLUME	metric_value_type	1	1	fcm_message_send_volume - FCM メッセージ送信ボリューム
FCM_MESSAGE_RECV_VOLUME	metric_value_type	1	1	fcm_message_recv_volume - FCM メッセージ受信ボリューム
FCM_TQ_SENDS_TOTAL	metric_value_type	1	1	fcm_tq_sends_total - FCM 表キュー送信の合計
FCM_TQ_RECVS_TOTAL	metric_value_type	1	1	fcm_tq_recvs_total - FCM 表キュー受信の合計
FCM_TQ_SEND_VOLUME	metric_value_type	1	1	fcm_tq_send_volume - FCM 表キュー送信ボリューム
FCM_TQ_RECV_VOLUME	metric_value_type	1	1	fcm_tq_recv_volume - FCM 表キュー受信ボリューム
TQ_TOT_SEND_SPILLS	metric_value_type	1	1	tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数
TCPIP_SEND_VOLUME	metric_value_type	1	1	tcpip_send_volume - TCP/IP 送信ボリューム
TCPIP_RECV_VOLUME	metric_value_type	1	1	tcpip_recv_volume - TCP/IP 受信ボリューム
IPC_SEND_VOLUME	metric_value_type	1	1	ipc_send_volume - プロセス間通信の送信ボリューム
IPC_RECV_VOLUME	metric_value_type	1	1	ipc_recv_volume - プロセス間通信の受信ボリューム
POST_THRESHOLD_SORTS	metric_value_type	1	1	post_threshold_sorts - ポストしきい値ソート
POST_SHRTHRESHOLD_SORTS	metric_value_type	1	1	post_shrthreshold_sorts - ポスト共有しきい値ソート
SORT_OVERFLOWS	metric_value_type	1	1	sort_overflows - ソート・オーバーフロー
AUDIT_EVENTS_TOTAL	metric_value_type	1	1	audit_events_total - 監査イベントの合計
TOTAL_RQST_MAPPED_IN	metric_value_type	0	1	total_rqst_mapped_in - マッピングの際の要求の合計
TOTAL_RQST_MAPPED_OUT	metric_value_type	0	1	total_rqst_mapped_out - マッピングの際に除外された要求の合計
ACT_REJECTED_TOTAL	metric_value_type	1	1	act_rejected_total - リジェクトされたアクティビティーの合計
ACT_ABORTED_TOTAL	metric_value_type	1	1	act_aborted_total - 異常終了したアクティビティーの合計
TOTAL_SORTS	metric_value_type	1	1	total_sorts - ソート合計

metric_value_type

メトリック時間およびカウンターの値のタイプ。

アクティビティー・イベント・モニターを使用したアクティビティー・モニター・エレメントのキャプチャー

アクティビティー・イベント・モニターには、event_activity 論理データ・グループの details_xml モニター・エレメントが含まれます。このモニター・エレメントを使用して、アクティビティーに関する情報をキャプチャーします。

モニター・エレメント **details_xml** は、MON_GET_ACTIVITY_DETAILS 表関数により報告されるすべてのアクティビティー・モニター・エレメントが含まれる XML 文書を返します。アクティビティー・モニター・エレメントは、MON_GET_ACTIVITY_DETAILS 表関数の DETAILS 列で報告される、アクティビティー詳細文書のサブセットです。

アクティビティー・モニター・エレメントは、ワークロードの COLLECT ACTIVITY METRICS 節、またはデータベース・レベルの **mon_act_metrics** データベース構成パラメーターにより制御されます。モニター・エレメントは、アクティビティーをサブミットする接続が、アクティビティー・モニター・エレメント収集が使用可能となっているワークロードまたはデータベースと関連付けられている場合に収集されます。アクティビティー・モニター・エレメントがアクティビティーについて収集されない場合、DETAILS_XML には空の文書が入ります。

DETAILS_XML 列で返される XML 文書のスキーマは、ファイル sqllib/misc/DB2MonCommon.xsd から入手可能です。最上位エレメントは、activity_metrics です。

activity_metrics モニター・エレメントの XML スキーマ

activity_metrics モニター・エレメントには、MON_GET_ACTIVITY_DETAILS 表関数によってレポートされるすべてのアクティビティー・メトリックが含まれます。

activty_metrics

アクティビティー・レベル・メトリック。

表 32. activty_metrics

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
activity_metrics	activity_level_metrics	1	1	アクティビティー・レベル・メトリック。

activity_level_metrics

このタイプは、アクティビティー・レベルの一部であるメトリックを定義します。

表 33. activity_level_metrics 属性

名前	データ・タイプ	使用	説明
release	xs:nonNegativeInteger	必須	このイベントがキャプチャーされた DB2 製品レベルを表します。

表 34. activity_level_metrics エレメント

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
WLM_QUEUE_TIME_TOTAL	metric_value_type	1	1	wlm_queue_time_total - ワークロード・マネージャー合計キュー時間
WLM_QUEUE_ ASSIGNMENTS_TOTAL	metric_value_type	1	1	wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て
FCM_TQ_RECV_WAIT_TIME	metric_value_type	1	1	fcm_tq_recv_wait_time - FCM 表キュー受信待機時間
FCM_MESSAGE_RECV_ WAIT_TIME	metric_value_type	1	1	fcm_message_recv_wait_time - FCM メッセージの受信待機時間
FCM_TQ_SEND_WAIT_TIME	metric_value_type	1	1	fcm_tq_send_wait_time - FCM 表キュー送信待機時間
FCM_MESSAGE_SEND_ WAIT_TIME	metric_value_type	1	1	fcm_message_send_wait_time - FCM メッセージの送信待機時間
LOCK_WAIT_TIME	metric_value_type	1	1	lock_wait_time - ロック待機中の時間
LOCK_WAITS	metric_value_type	1	1	lock_waits - ロック待機数
DIRECT_READ_TIME	metric_value_type	1	1	direct_read_time - 直接読み取り時間
DIRECT_READ_REQS	metric_value_type	1	1	direct_read_reqs - 直接読み取り要求
DIRECT_WRITE_TIME	metric_value_type	1	1	direct_write_time - 直接書き込み時間
DIRECT_WRITE_REQS	metric_value_type	1	1	direct_write_reqs - 直接書き込み要求
LOG_BUFFER_WAIT_TIME	metric_value_type	1	1	log_buffer_wait_time - ログ・バッファー待機時間
NUM_LOG_BUFFER_FULL	metric_value_type	1	1	num_log_buffer_full - フル・ログ・バッファーの回数
LOG_DISK_WAIT_TIME	metric_value_type	1	1	log_disk_wait_time - ログ・ディスク待機時間
LOG_DISK_WAITS_TOTAL	metric_value_type	1	1	log_disk_waits_total - ログ・ディスク待機の合計
POOL_WRITE_TIME	metric_value_type	1	1	pool_write_time - バッファー・プール物理書き込み時間の合計
POOL_READ_TIME	metric_value_type	1	1	pool_read_time - バッファー・プール物理読み取り時間の合計
AUDIT_FILE_WRITE_ WAIT_TIME	metric_value_type	1	1	audit_file_write_wait_time - 監査ファイル書き込み待機時間
AUDIT_FILE_WRITES_TOTAL	metric_value_type	1	1	audit_file_writes_total - 書き込まれた監査ファイルの合計

表 34. activity_level_metrics エレメント (続き)

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
AUDIT_SUBSYSTEM_ WAIT_TIME	metric_value_type	1	1	audit_subsystem_wait_time - 監査サブシステム待機時間
AUDIT_SUBSYSTEM_ WAITS_TOTAL	metric_value_type	1	1	audit_subsystem_waits_total - 監査サブシステム待機の合計
DIAGLOG_WRITE_WAIT_ TIME	metric_value_type	1	1	diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間
DIAGLOG_WRITES_TOTAL	metric_value_type	1	1	diaglog_writes_total - 診断ログ・ファイル書き込みの合計
FCM_SEND_WAIT_TIME	metric_value_type	1	1	fcm_send_wait_time - FCM 送信待機時間
FCM_RECV_WAIT_TIME	metric_value_type	1	1	fcm_recv_wait_time - FCM 受信待機時間
TOTAL_ACT_WAIT_TIME	metric_value_type	1	1	total_act_wait_time - 合計アクティビティー待機時間
TOTAL_SECTION_SORT_ PROC_TIME	metric_value_type	1	1	total_section_sort_proc_time - セクション・ソート処理時間の合計
TOTAL_SECTION_SORT_ TIME	metric_value_type	1	1	total_section_sort_time - セクション・ソート時間の合計
TOTAL_SECTION_SORTS	metric_value_type	1	1	total_section_sorts - セクション・ソートの合計
TOTAL_ACT_TIME	metric_value_type	1	1	total_act_time - 合計アクティビティー時間
ROWS_READ	metric_value_type	1	1	rows_read - 読み取り行数
ROWS_MODIFIED	metric_value_type	1	1	rows_modified - 変更行数
POOL_DATA_L_READS	metric_value_type	1	1	pool_data_l_reads - バッファー・プール・データの論理読み取り
POOL_INDEX_L_READS	metric_value_type	1	1	pool_index_l_reads - バッファー・プール索引の論理読み取り
POOL_TEMP_DATA_ L_READS	metric_value_type	1	1	pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り
POOL_TEMP_INDEX_ L_READS	metric_value_type	1	1	pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り
POOL_XDA_L_READS	metric_value_type	1	1	pool_xda_l_reads - バッファー・プール XDA データの論理読み取り
POOL_TEMP_XDA_L_READS	metric_value_type	1	1	pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り
TOTAL_CPU_TIME	metric_value_type	1	1	total_cpu_time - 合計 CPU 時間
POOL_DATA_P_READS	metric_value_type	1	1	pool_data_p_reads - バッファー・プール・データの物理読み取り

表 34. activity_level_metrics エレメント (続き)

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
POOL_TEMP_DATA_P_READS	metric_value_type	1	1	pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り
POOL_XDA_P_READS	metric_value_type	1	1	pool_xda_p_reads - バッファー・プール XDA データの物理読み取り
POOL_TEMP_XDA_P_READS	metric_value_type	1	1	pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り
POOL_INDEX_P_READS	metric_value_type	1	1	pool_index_p_reads - バッファー・プール索引の物理読み取り
POOL_TEMP_INDEX_P_READS	metric_value_type	1	1	pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り
POOL_DATA_WRITES	metric_value_type	1	1	pool_data_writes - バッファー・プールへのデータの書き込み
POOL_XDA_WRITES	metric_value_type	1	1	pool_xda_writes - バッファー・プール XDA データの書き込み
POOL_INDEX_WRITES	metric_value_type	1	1	pool_index_writes - バッファー・プール索引の書き込み
DIRECT_READS	metric_value_type	1	1	direct_reads - データベースからの直接読み取り
DIRECT_WRITES	metric_value_type	1	1	direct_writes - データベースへの直接書き込み
ROWS_RETURNED	metric_value_type	1	1	rows_returned - 戻り行数
DEADLOCKS	metric_value_type	1	1	deadlocks - デッドロック検出数
LOCK_TIMEOUTS	metric_value_type	1	1	lock_timeouts - ロック・タイムアウト数
LOCK_ESCALS	metric_value_type	1	1	lock_escals - ロック・エスカレーション数
FCM_SENDS_TOTAL	metric_value_type	1	1	fcm_sends_total - FCM 送信の合計
FCM_RECVS_TOTAL	metric_value_type	1	1	fcm_recvs_total - FCM 受信の合計
FCM_SEND_VOLUME	metric_value_type	1	1	fcm_send_volume - FCM 送信ボリューム
FCM_RECV_VOLUME	metric_value_type	1	1	fcm_recv_volume - FCM 受信ボリューム
FCM_MESSAGE_SENDS_TOTAL	metric_value_type	1	1	fcm_message_sends_total - FCM メッセージ送信の合計
FCM_MESSAGE_RECVS_TOTAL	metric_value_type	1	1	fcm_message_recvs_total - FCM メッセージ受信の合計
FCM_MESSAGE_SEND_VOLUME	metric_value_type	1	1	fcm_message_send_volume - FCM メッセージ送信ボリューム

表 34. activity_level_metrics エレメント (続き)

名前	データ・タイプ	最小オカレンス	最大オカレンス	説明
FCM_MESSAGE_RECV_VOLUME	metric_value_type	1	1	fcm_message_recv_volume - FCM メッセージ受信ボリューム
FCM_TQ_SENDS_TOTAL	metric_value_type	1	1	fcm_tq_sends_total - FCM 表キュー送信の合計
FCM_TQ_RECVS_TOTAL	metric_value_type	1	1	fcm_tq_recvs_total - FCM 表キュー受信の合計
FCM_TQ_SEND_VOLUME	metric_value_type	1	1	fcm_tq_send_volume - FCM 表キュー送信ボリューム
FCM_TQ_RECV_VOLUME	metric_value_type	1	1	fcm_tq_recv_volume - FCM 表キュー受信ボリューム
TQ_TOT_SEND_SPILLS	metric_value_type	1	1	tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数
POST_THRESHOLD_SORTS	metric_value_type	1	1	post_threshold_sorts - ポストしきい値ソート
POST_SHRTHRESHOLD_SORTS	metric_value_type	1	1	post_shrthreshold_sorts - ポスト共有しきい値ソート
SORT_OVERFLOWS	metric_value_type	1	1	sort_overflows - ソート・オーバーフロー
AUDIT_EVENTS_TOTAL	metric_value_type	1	1	audit_events_total - 監査イベントの合計
TOTAL_SORTS	metric_value_type	1	1	total_sorts - ソート合計

metric_value_type

メトリック時間およびカウンターの値のタイプ。

表、ファイル、およびパイプに書き込むイベント・モニター

特定のイベント・モニターを構成して、表、パイプ、またはファイルにデータベース・イベントを書き込むことができます。

イベント・モニターを使用して、指定されたイベントの発生時に、データベースおよび接続されたアプリケーションに関する情報を収集します。イベントは、接続、デッドロック、ステートメント、トランザクションなどの、データベース・アクティビティーの遷移を表します。モニターするイベント (1 つ以上) のタイプごとにイベント・モニターを定義することができます。例えば、デッドロック・イベント・モニターは、デッドロックが発生するのを待機します。発生すると、関係するアプリケーションおよび競合するロックに関する情報を収集します。

イベント・モニターを作成するには、CREATE EVENT MONITOR SQL ステートメントを使用します。イベント・モニターは、それらがアクティブなときにだけイベント・データを収集します。イベント・モニターを活動化または非活動化するには、 SET EVENT MONITOR STATE SQL ステートメントを使用します。イベント・モニターの状況 (アクティブか非アクティブか) は、 SQL 関数 EVENT_MON_STATE によって判別することができます。

CREATE EVENT MONITOR SQL ステートメントを実行すると、それが作成するイベント・モニターの定義が、以下のデータベース・システム・カタログ表に保管されます。

- SYSCAT.EVENTMONITORS: データベースについて定義されたイベント・モニター
- SYSCAT.EVENTS: データベースについてモニターされるイベント
- SYSCAT.EVENTTABLES: 表イベント・モニターのためのターゲット表

それぞれのイベント・モニターには、モニター・エレメント内のインスタンスのデータの、独自の専用論理ビューがあります。特定のイベント・モニターが非活動化された後、再活動化されると、これらのカウンターのビューがリセットされます。リセットは、新たに活動化されたイベント・モニターだけで行われます。他のすべてのイベント・モニターは、引き続きカウンター値の独自のビューを使用し続けます (追加があればそのカウンター値に追加します)。

イベント・モニターの出力は、非パーティション SQL 表、ファイル、または Named PIPE に送ることができます。

注: デフォルトでデータベースごとに、推奨されなくなった詳細なデッドロック・イベント・モニター DB2DETAILDEADLOCK が作成され、データベースが活動化される時にそれが開始します。 このイベント・モニターによるオーバーヘッドは、これをドロップすることによって回避してください。DB2DETAILDEADLOCK モニター・エレメントの使用は推奨されていません。 この推奨されていないイベント・モニターは将来のリリースで除去される可能性があります。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

データベース・システム・イベントからの情報の収集

イベント・モニターを使用して、指定されたイベントの発生時に、データベースおよび接続されたアプリケーションに関する情報を収集します。イベント・モニターはデータベース・オブジェクトであり、 SQL データ定義言語 (SQL DDL) ステートメントを使用して作成され、操作されます。

イベント・モニターを作成し、操作するには、SQLADM または DBADM 権限が必要です。

ある特定のタイプのイベント・モニターでは、出力の送信先をファイル、パイプ、もしくは (通常) 表から選択できるようになっています。 このタイプのモニターに関する説明をここに記します。 説明の一部は、フォーマットされていないイベント表へ出力を送信するイベント・モニターに適用できないものがあります。

以下にリストするステップは、イベント・モニターの典型的なライフ・サイクルを表しています。これらのステップは、必ずしも示されている順序で実行する必要があるとは限りません。例えば、使用法に応じて、イベント・モニターがドロップされなかったり、非活動化されない場合もあります。しかし、イベント・モニターのライフ・サイクルでは、以下の 2 つの事柄は必ず行われます。つまり、最初のステップは常にイベント・モニターの作成で、最後のステップは常にイベント・モニターの削除です。

1. 68 ページの『イベント・モニターの作成』。
2. *ファイルおよびパイプ・イベント・モニターの場合のみ:*
   - イベント・レコードを受け取るディレクトリーまたは Named PIPE が存在することを確認します。存在しない場合は、イベント・モニターは活動化されません。

     AIX® では、mkfifo コマンドを使用して Named PIPE を作成することができます。 Linux® および他の UNIX® タイプ (Solaris オペレーティング・システムなど) では、pipe() ルーチンを使用します。

     Windows® では、CreateNamedPipe() ルーチンを使用して Named PIPE を作成することができます。
   - *パイプ・イベント・モニターの場合のみ:* イベント・モニターを活動化する前に Named PIPE を開きます。これは、次のオペレーティング・システムの機能を使って行うことができます。
     - UNIX の場合: `open()`
     - Windows の場合: `ConnectNamedPipe()`

     また、次のように db2evmon 実行可能プログラムを使って行うこともできます。

     ```
     db2evmon -db databasename
              -evm eventmonname
     ```

     `databasename` は、モニター対象のデータベースの名前を表します。

     `evmonname` は、イベント・モニターの名前を表します。
3. 新しく作成したイベント・モニターを活動化して、それが情報を収集できるようにします。

   ```
   SET EVENT MONITOR evmonname STATE 1;
   ```

   AUTOSTART オプションを指定してイベント・モニターを作成すると、最初のユーザーがデータベースに接続する際にイベント・モニターは活動化されます。さらにイベント・モニターは、いったん明示的に活動化されると、データベースが再活動化される時に自動的に再始動します。この再始動は、イベント・モニターが明示的に非活動にされるか、インスタンスが停止するまで行われます。 表イベント・モニターを開始すると、イベント・モニターによって SYSCAT.EVENTMONITORS カタログ表の evmon_activates 列が更新されます。この変更はログに記録されるため、DATABASE CONFIGURATION によって次が表示されます。

   ```
   Database is consistent                                  = NO
   ```

   イベント・モニターが AUTOSTART オプションを使用して作成されている場合、最初のユーザーがデータベースに接続してデータベースを非活動化するために即時に切断すると、ログ・ファイルが作成されます。
4. イベント・モニターがアクティブか非アクティブかを調べるために、次のように表 SYSCAT.EVENTMONITORS に対する照会で SQL 関数 EVENT_MON_STATE を発行します。

   ```
   SELECT evmonname, EVENT_MON_STATE(evmonname) FROM
          syscat.eventmonitors;
   ```

存在するすべてのイベント・モニターのリストとそれらの状況が表示されます。戻り値 0 は、指定されたイベント・モニターが非アクティブであり、 1 は、アクティブであることを示します。

5. イベント・モニター出力を読み取ります。表書き込みイベント・モニターの場合はターゲット表で確認する必要があります。 CLP からファイルまたはパイプ・イベント・モニターのデータにアクセスするには、関連タスク『コマンド行からのファイルまたはパイプ・イベント・モニター出力のフォーマット』を参照してください。
6. イベント・モニターを非活動化する、つまりイベント・モニターを OFF にするには、次のように SET EVENT MONITOR ステートメントを使用します。

   ```
   SET EVENT MONITOR evmonname STATE 0
   ```

   イベント・モニターを非活動化しても、それが削除されることはありません。イベント・モニターは休止データベース・オブジェクトとして存在します。イベント・モニターを非活動化すると、その内容がすべてフラッシュされます。そのため、非活動化されているイベント・モニターを再活動化する場合には、再活動化以降に収集された情報だけが含まれます。

   データベースが非活動化するときに、アクティビティー・イベント・モニターがアクティブである場合、キューにあるバックログされたアクティビティー・レコードはすべて廃棄されるので注意してください。確実にすべてのアクティビティー・イベント・モニター・レコードを入手し、何も廃棄されないようにするには、データベースを非活動化する前に、まずアクティビティー・イベント・モニターを明示的に非活動化します。アクティビティー・イベント・モニターを明示的に非活動化した場合、キューにあるバックログされたアクティビティー・レコードはすべて、イベント・モニターが非活動化される前に処理されます。
7. パイプ・イベント・モニターを非活動化した後、対応する Named PIPE を閉じます。 UNIX では `close()` 関数を使用し、Windows 2000 では `DisconnectNamedPipe()` 関数を使用します。
8. イベント・モニター・オブジェクトをドロップするには、次のように DROP EVENT MONITOR ステートメントを使用します。

   ```
   DROP EVENT MONITOR evmonname
   ```

   イベント・モニターが非アクティブの場合にのみ、それをドロップできます。
9. パイプ・イベント・モニターをドロップした後、対応する Named PIPE を削除します。 UNIX では `unlink()` 関数を使用し、Windows では `CloseHandle()` 関数を使用します。 表書き込みイベント・モニターをドロップするときに、関連したターゲット表はドロップされません。同様に、ファイル・イベント・モニターをドロップするときに、関連したファイルは削除されません。

イベント・モニターの作成

イベント・モニターのライフ・サイクルにおける最初のステップは、イベント・モニターの作成です。イベント・モニターを作成する前に、イベント・レコードの宛先を決定する必要があります。それは、SQL 表、ファイル、または Named PIPE を介して送信します。

イベント・モニターを作成するには、SQLADM または DBADM 権限が必要です。

それぞれのイベント・レコードの宛先ごとに、CREATE EVENT MONITOR SQL ステートメントで指定される特定のオプションがあります。CREATE EVENT MONITOR ステートメントのターゲット表は、パーティション表以外の表でなければなりません。パーティション・データベースでのイベントのモニターでは特別な注意も必要です。

ある特定のタイプのイベント・モニターでは、出力の送信先をファイル、パイプ、もしくは (通常) 表から選択できるようになっています。 このタイプのモニターに関する説明をここに記します。 説明の一部は、フォーマットされていないイベント表へ出力を送信するイベント・モニターに適用できないものがあります。

1. 表イベント・モニターの作成。
2. ファイル・イベント・モニターの作成。
3. パイプ・イベント・モニターの作成。
4. パーティション・データベース用のイベント・モニターの作成。

イベント・モニターが作成されて活動化されると、指定されたイベントが発生するたびに、モニター・データを記録します。

表イベント・モニターの作成

イベント・モニターの作成時に、それが収集した情報の保管場所を決定する必要があります。表イベント・モニターは SQL 表にイベント・レコードを流します。これは、ファイルおよびパイプ・イベント・モニターの簡単な代替手段で、イベント・モニター・データのキャプチャー、解析、および管理を、より簡単に行えるようにします。イベント・モニターが収集するすべてのイベント・タイプで、関連するそれぞれの論理データ・グループについてのターゲット表が作成されます。

表イベント・モニターを作成するには、SQLADM または DBADM 権限が必要です。

CREATE EVENT MONITOR ステートメントのターゲット表は、パーティション表以外の表でなければなりません。

表イベント・モニターの各種オプションは、CREATE EVENT MONITOR ステートメントで設定されます。表書き込みイベント・モニター用の CREATE EVENT MONITOR SQL ステートメントの生成をより簡単に行うために、 db2evtbl コマンドを使用することができます。イベント・モニターの名前および目的のイベント・タイプ (1 つ以上) を指定するだけで、 CREATE EVENT MONITOR ステートメントが生成され、すべてのターゲット表のリストが完成します。次に、生成されたステートメントをコピーして、変更を加えれば、 CLP からステートメントを実行することができます。

1. イベント・モニター・データが表 (または表のセット) に収集されることを指定します。

   ```
   CREATE EVENT MONITOR dlmon FOR eventtype
                       WRITE TO TABLE
   ```

   dlmon はイベント・モニターの名前です。
2. モニター対象のイベントのタイプを指定します。単一イベント・モニターで、1 つ以上のイベント・タイプをモニターすることができます。

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                    WRITE TO TABLE
```

このイベント・モニターは、CONNECTIONS および DEADLOCKS WITH DETAILS イベント・タイプをモニターします。上記のステートメントがユーザー 'riihi' によって発行されたとすると、ターゲット表の派生名および表スペースが以下のようになります。

- riihi.connheader_dlmon
- riihi.conn_dlmon
- riihi.connmemuse_dlmon
- riihi.deadlock_dlmon
- riihi.dlconn_dlmon
- riihi.dllock_dlmon
- riihi.control_dlmon

3. BUFFERSIZE 値を調整することによって、表イベント・モニター・バッファーのサイズを指定します (4K ページ単位)。

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                    WRITE TO TABLE BUFFERSIZE 8
```

8 は、2 つのイベント表バッファーを合わせた容量です (4K ページ単位)。それぞれのバッファーが 16K で、最大 32K までバッファー・スペースが追加されます。

個々のバッファーのデフォルト・サイズは 4 ページです (16K のバッファーが 2 つ割り振られます)。最小サイズは 1 ページです。バッファーはモニター・ヒープから割り振られるので、バッファーの最大サイズはこのヒープのサイズによって制限されます。パフォーマンス上の理由で、アクティブ率の高いイベント・モニターには、アクティブ率が比較的低いものよりも大きなバッファーが必要です。

4. イベント・モニターをブロック化または非ブロック化する必要があるかどうかを指定します。ブロック化イベント・モニターの場合、イベントを生成する各エージェントは、イベント・バッファーがいっぱいであれば、それが表に書き込まれるまで待機します。これは、データベースのパフォーマンスを低下させることがあります。なぜなら、バッファーがクリアされるまで、延期されたエージェントおよび従属エージェントが実行できなくなるからです。イベント・データが脱落しないようにするには、BLOCKED 節を使用します。

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                    WRITE TO TABLE BUFFERSIZE 8 BLOCKED
```

イベント・モニターはデフォルトではブロック化されます。

すべての単一イベント・レコードを収集することよりも、データベース・パフォーマンスのほうが重要な場合には、非ブロック化イベント・モニターを使用してください。その場合、イベントを生成する各エージェントは、イベント・バッファーがいっぱいのときに、それが表に書き込まれるのを待機しません。結果として、非ブロック化イベント・モニターは、アクティブ率の高いシステ

ムではデータの脱落という影響を受けます。イベント・モニターによって生じるパフォーマンス上のオーバーヘッドを最小限に抑えるには、NONBLOCKED 節を使用します。

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                    WRITE TO TABLE BUFFERSIZE 8 NONBLOCKED
```

5. イベント・レコードの収集元の論理データ・グループを指定します。イベント・モニターはそれぞれの論理データ・グループからのデータを、対応する表に保管します。

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                    WRITE TO TABLE CONN, DLCONN, DLLOCK
                    BUFFERSIZE 8 NONBLOCKED
```

CONN、DLCONN、および DLLOCK 論理データ・グループが選択されます。その他の選択可能な論理データ・グループ CONNHEADER、DEADLOCK、または CONTROL については言及されていません。
CONNHEADER、DEADLOCK、または CONTROL に関係するデータは、dlmon イベント・モニターには保管されません。

6. データを収集する必要のあるモニター・エレメントを指定します。

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
             WRITE TO TABLE CONN,
             DLCONN (EXCLUDES(agent_id, lock_wait_start_time)),
             DLLOCK (INCLUDES(lock_mode, table_name))
             BUFFERSIZE 8 NONBLOCKED
```

CONN のすべてのモニター・エレメントがキャプチャーされます (これがデフォルトの動作です)。DLCONN の場合は、**agent_id** および **lock_wait_start_time** 以外のすべてのモニター・エレメントがキャプチャーされます。最後に、DLLOCK の場合は、モニター・エレメント **lock_mode** および **table_name** だけがキャプチャーされます。

7. 作成される表の名前を指定し、表スペースを指定します。

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
WRITE TO TABLE CONN,
DLCONN (TABLE mydept.dlconnections
EXCLUDES(agent_id, lock_wait_start_time)),
DLLOCK (TABLE dllocks IN mytablespace
INCLUDES(lock_mode, table_name))
BUFFERSIZE 8 NONBLOCKED
```

上記のステートメントがユーザー riihi によって発行されたとすると、ターゲット表の派生名および表スペースが以下のようになります。

- CONN: riihi.conn_dlmon (デフォルトの表スペースで)
- DLCONN: mydept.dlconnections (デフォルトの表スペースで)
- DLLOCK: riihi.dllocks (MYTABLESPACE 表スペースで)

デフォルトの表スペースは、イベント・モニター定義プログラムに USE 特権があれば、 IBMDEFAULTGROUP から割り当てられます。定義プログラムがこの表スペースに対する USE 特権を有していない場合には、定義プログラムが USE 特権を有する表スペースが割り当てられます。

8. 表スペースがどの程度いっぱいになれば、イベント・モニターが自動的に非活動化するのかを指定します。

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                    WRITE TO TABLE DLCONN PCTDEACTIVATE 90
                    BUFFERSIZE 8 NONBLOCKED
```

表スペースが 90% の容量に達すれば、dlmon イベント・モニターが自動的にシャットオフします。DMS 表スペースには PCTDEACTIVATE 節のみを使用できます。ターゲット表スペースで自動サイズ変更が有効な場合は、PCTDEACTIVATE 文節を 100 に設定してください。

9. データベースを開始するたびに、イベント・モニターが自動的に活動化されるかどうかを指定します。デフォルトでは、データベースの開始時にイベント・モニター (WLM イベント・モニターは例外) は自動的には活動化されません。
   - データベースの開始時に自動的に開始するイベント・モニターを作成するには、次のステートメントを発行する。

     ```
     CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                         WRITE TO TABLE BUFFERSIZE 8 NONBLOCKED
                         AUTOSTART NONBLOCKED
     ```

   - データベースの開始時に自動的に開始しないイベント・モニターを作成するには、次のステートメントを発行する。

     ```
     CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                         WRITE TO TABLE BUFFERSIZE 8 NONBLOCKED
                         MANUALSTART
     ```

10. イベント・モニターを活動化または非活動化するには、SET EVENT MONITOR STATE ステートメントを使用します。

表イベント・モニターが作成されて活動化されると、指定されたイベントが発生するたびに、モニター・データを記録します。

イベント・モニターの表管理

SQL 表にイベント・レコードを保管するように、イベント・モニターを定義することができます。これを行うには、CREATE EVENT MONITOR ステートメントを WRITE TO TABLE 節を付けて使用します。

表書き込みイベント・モニターを作成すると、データベースは、データを戻す論理データ・グループのそれぞれについて、レコードを保管するためのターゲット表 を作成します。デフォルトでは、データベースはイベント・モニターの作成プログラムのスキーマで表を作成し、それに対応する論理データ・グループおよびイベント・モニター名に従って表に名前を付けます。それぞれの表において、列名はそれが表すモニター・エレメント名と一致しています。

例えば、ユーザー riihi が STATEMENTS イベントをキャプチャーするイベント・モニターを作成しているとします。

```
CREATE EVENT MONITOR foo FOR STATEMENTS WRITE TO TABLE
```

STATEMENTS イベント・タイプを使用するイベント・モニターは、event_connheader、event_stmt、および event_subsection 論理データ・グループからデータを収集します。データベースは以下の表を作成しました。

- riihi.connheader_foo
- riihi.stmt_foo
- riihi.subsection_foo

- riihi.control_foo

個々のイベント・タイプに固有な論理データ・グループを表す表に加えて、すべての表書き込みイベント・モニターに関するコントロール表が 1 つ作成されます。この表は上記の riihi.control_foo のことです。コントロール表には、イベント・モニター・メタデータ、特に event_start、event_db_header (**conn_time** モニター・エレメントのみ)、および event_overflow 論理データ・グループからのメタデータが含まれています。

表書き込みイベント・モニターが活動状態になると、各ターゲット表の IN 表ロックを獲得します。これにより、このイベント・モニターが活動状態の間はそれらの表が変更されないようになります。この表ロックは、このイベント・モニターが活動状態の間はすべての表で保持されます。何らかのターゲット表で排他的アクセスが必要な場合 (ユーティリティーを実行する場合など)、そのようなアクセスを試みる前に、まずこのイベント・モニターを非活動化して、表ロックを解放してください。

ターゲット表の各列名は、イベント・モニターのエレメント ID と一致しています。対応するターゲット表の列がないイベント・モニター・エレメントは無視されます。

表書き込みイベント・モニターのターゲット表 (未フォーマット・イベント (UE) 表を含む) は、手動で整理する必要があります。 アクティブ量の非常に多いシステムでは、イベント・モニターが大容量のデータを記録するため、システム・リソースをすぐに使い尽くしてしまうことがあります。ファイルや Named PIPE に書き込むイベント・モニターとは異なり、表書き込みイベント・モニターは、特定の論理データ・グループまたはモニター・エレメントだけを記録するように定義することができます。このフィーチャーにより、ユーザーの目的にかなうデータだけを収集することができ、イベント・モニターによって生成されるデータのボリュームを削減することができます。例えば、次のステートメントは、 event_xact 論理データ・グループだけから TRANSACTIONS イベントをキャプチャーし、 **lock_escal** モニター・エレメントだけを含めるように、イベント・モニターを定義します。

```
CREATE EVENT MONITOR foo_lite FOR TRANSACTIONS WRITE TO TABLE
                    XACT(INCLUDES(lock_escal))
```

注: このオプションは推奨されなくなりました。 このオプションの使用は推奨されておらず、将来のリリースでは除去される予定です。 CREATE EVENT MONITOR FOR UNIT OF WORK ステートメントを使用して、トランザクション・イベントをモニターしてください。

イベント・モニターのターゲット表をデフォルトの表スペース内のデフォルトのスキーマに、デフォルトの表名で常駐させるのが望ましくない場合があります。例えば、モニター・データのボリュームが大きくなることが予想されるのであれば、ターゲット表を独自の表スペースに配置することもできます。

スキーマ、表、および表スペース名は CREATE EVENT MONITOR ステートメントで指定することができます。スキーマ名は表名とともに提供され、表の派生名を形成します。

ターゲット表を使用できるのは、単一イベント・モニターに限られます。ターゲット表がすでに別のイベント・モニターについて定義されているのが検出されたか、または他の理由で作成できない場合には、CREATE EVENT MONITOR ステートメントは失敗します。

表スペース名は表名の後に、オプションの IN 節を付けて追加することができます。 DB2 が自動的に作成するターゲット表とは異なり、表スペースがイベント・モニター定義に組み込まれている場合には、それがすでに存在していなければなりません。表スペースが指定されていない場合には、定義プログラムが USE 特権を有する表スペースが割り当てられます。

パーティション・データベース環境では、表書き込みイベント・モニターは、イベント・モニター表を含む表スペースが存在するデータベース・パーティション上でのみアクティブになります。特定のデータベース・パーティション上にアクティブ・イベント・モニターのターゲット表スペースが存在しない場合、そのデータベース・パーティションのイベント・モニターは非活動状態にされ、db2diag ログ・ファイルにエラーが書き込まれます。

イベント・モニター・データの検索時のパフォーマンスを向上させるために、イベント表に索引を作成することができます。また、トリガー、関係保全、制約など、他の表属性を追加することもできます。イベント・モニターはそれらを無視します。

例えば、次のステートメントは、event_connheader、event_stmt、および event_subsection 論理データ・グループを使用して、 STATEMENTS イベントをキャプチャーするイベント・モニターを定義します。 3 つのターゲット表のそれぞれには、異なるスキーマ、表、および表スペースの組み合わせがあります。

```
CREATE EVENT MONITOR foo FOR STATEMENTS
WRITE TO TABLE CONNHEADER,
STMT (TABLE mydept.statements),
SUBSECTION (TABLE subsections, IN mytablespace)
```

上記のステートメントがユーザー 'riihi' によって発行されたとすると、ターゲット表の派生名および表スペースが以下のようになります。

- CONNHEADER: riihi.connheader_foo (デフォルトの表スペースで)
- STMT: mydept.statements (デフォルトの表スペースで)
- SUBSECTION: riihi.subsections (MYTABLESPACE 表スペースで)

イベント・モニターを活動化しているときにターゲット表が存在しない場合でも、活動化は続行しますが、ターゲット表に挿入されるはずだったデータは無視されます。また、ターゲット表にモニター・エレメント専用の列がない場合にも、そのモニター・エレメントは無視されます。

表書き込みイベント・モニターがアクティブな場合、イベント・レコードを保管する表スペースがその限界に達する可能性があります。 DMS 表スペースについてこのリスクを制御するために、イベント・モニターが非活動化されるときの表スペース容量のパーセンテージを定義することができます。これは、CREATE EVENT MONITOR ステートメントの PCTDEACTIVATE 節で宣言することができます。

SMS 表スペースの場合、この値は 100 に設定されます。ターゲット表スペースで自動サイズ変更が有効な場合は、PCTDEACTIVATE を 100 に設定するようにお勧めします。

非パーティションのデータベース環境では、最後のアプリケーションが終了すると(それまでにデータベースが明示的に活動化されていないと)、表イベント・モニターへの書き込みはすべて非活動化されます。パーティション・データベース環境では、カタログ・パーティションが非活動化されると表イベント・モニターへの書き込みが非活動化されます。

次の表は、ターゲット表が戻されるイベント・タイプごとの、デフォルトのターゲット表名を示しています。

表 35. 表書き込みイベント・モニターのターゲット表

イベント・タイプ	ターゲット表名	入手できる情報
DEADLOCKS[1]	CONNHEADER	接続メタデータ
	DEADLOCK	デッドロック・データ
	DLCONN	デッドロックに関係しているアプリケーションとロック
	CONTROL	イベント・モニター・メタデータ
DEADLOCKS WITH DETAILS[1]	CONNHEADER	接続メタデータ
	DEADLOCK	デッドロック・データ
	DLCONN	デッドロックに関係しているアプリケーション
	DLLOCK	デッドロックに関係しているロック
	CONTROL	イベント・モニター・メタデータ
DEADLOCKS WITH DETAILS HISTORY[1]	CONNHEADER	接続メタデータ
	DEADLOCK	デッドロック・データ
	DLCONN	デッドロックに関係しているアプリケーション
	DLLOCK	デッドロックに関係しているロック
	STMTHIST	作業単位内の以前のステートメントのリスト
	CONTROL	イベント・モニター・メタデータ
DEADLOCKS WITH DETAILS HISTORY VALUES[1]	CONNHEADER	接続メタデータ
	DEADLOCK	デッドロック・データ
	DLCONN	デッドロックに関係しているアプリケーション
	DLLOCK	デッドロックに関係しているロック
	STMTHIST	作業単位内の以前のステートメントのリスト
	STMTVALS	STMTHIST 表内のステートメントの入力データ値
	CONTROL	イベント・モニター・メタデータ
STATEMENT	CONNHEADER	接続メタデータ
	STMT	ステートメント・データ
	SUBSECTION	サブセクションに固有のステートメント・データ
	CONTROL	イベント・モニター・メタデータ
TRANSACTIONS[2]	CONNHEADER	接続メタデータ
	XACT	トランザクション・データ
	CONTROL	イベント・モニター・メタデータ

表 35. 表書き込みイベント・モニターのターゲット表 (続き)

イベント・タイプ	ターゲット表名	入手できる情報
CONNECTIONS	CONNHEADER	接続メタデータ
	CONN	接続データ
	CONTROL	イベント・モニター・メタデータ
	CONNMEMUSE	メモリー・プール・メタデータ
DATABASE	DB	データベース・マネージャー・データ
	CONTROL	イベント・モニター・メタデータ
	DBMEMUSE	メモリー・プール・メタデータ
BUFFERPOOLS	BUFFERPOOL	バッファー・プール・データ
	CONTROL	イベント・モニター・メタデータ
TABLESPACES	TABLESPACE	表スペース・データ
	CONTROL	イベント・モニター・メタデータ
TABLES	TABLE	表データ
	CONTROL	イベント・モニター・メタデータ
ACTIVITIES	ACTIVITY	実行が完了した、または進行中にキャプチャーされたアクティビティー。
	ACTIVITYSTMT	ステートメントであるアクティビティーに関するステートメント情報。
	ACTIVITYVALS	アクティビティーの入力データ値。 CLOB、REF、BOOLEAN、STRUCT、DATALINK、LONG VARGRAPHIC、LONG、XMLLOB、 DBCLOB 以外のデータ・タイプについてレポートを作成できます。
	CONTROL	イベント・モニター・メタデータ
STATISTICS	SCSTATS	システム内のそれぞれのサービス・クラス、処理クラス、またはワークロードで実行されたアクティビティーから算出される統計。
	WCSTATS	
	WLSTATS	
	HISTOGRAMBIN	
	QSTATS	
	CONTROL	イベント・モニター・メタデータ
THRESHOLD VIOLATIONS	THRESHOLDVIOLATIONS	違反したしきい値とその時間についてのリスト。
	CONTROL	イベント・モニター・メタデータ

[1] このオプションは推奨されなくなりました。 このオプションの使用は推奨されておらず、将来のリリースでは除去される予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

[2] このオプションは推奨されなくなりました。 このオプションの使用は推奨されておらず、将来のリリースでは除去される予定です。 CREATE EVENT MONITOR FOR UNIT OF WORK ステートメントを使用して、トランザクション・イベントをモニターしてください。

次の論理データ・グループは、表書き込みイベント・モニターでは収集されません。

- event_log_stream_header
- event_log_header
- event_dbheader (**conn_time** モニター・エレメントだけが収集される)

イベント・モニター表の各列のデータ・タイプは、列によって表されるモニター・エレメントのデータ・タイプに対応します。以下は、モニター・エレメントの元のシステム・モニター・データ・タイプ (sqlmon.h にある) を、表列の SQL データ・タイプに対応させた、データ・タイプのマッピングの集合です。

表 36. システム・モニター・データ・タイプのマッピング

システム・モニターのデータ・タイプ	SQL データ・タイプ
SQLM_TYPE_STRING	CHAR[n]、VARCHAR[n]、CLOB[n]
SQLM_TYPE_U8BIT および SQLM_TYPE_8BIT	SMALLINT、INTEGER、または BIGINT
SQLM_TYPE_U16BIT および SQLM_TYPE_16BIT	SMALLINT、INTEGER、または BIGINT
SQLM_TYPE_U32BIT および SQLM_TYPE_32BIT	INTEGER または BIGINT
SQLM_TYPE_U64BIT および SQLM_TYPE_64BIT	BIGINT
SQLM_TIMESTAMP	TIMESTAMP
SQLM_TIME	BIGINT
SQLCA: SQLERRMC	VARCHAR[72]
SQLCA: SQLSTATE	CHAR[5]
SQLCA: SQLWARN	CHAR[11]
SQLCA: 他のフィールド	INTEGER または BIGINT
SQLM_TYPE_HANDLE	BLOB[n]

注:

1. すべて列は NOT NULL です。
2. CLOB 列がある表のパフォーマンスは VARCHAR 列がある表より劣るため、stmt evmGroup (またはデッドロックの詳細を使用するときに evmGroup) を指定するときは、 TRUNC キーワードを使用することを検討してください。
3. SQLM_TYPE_HANDLE は、コンパイル環境ハンドル・オブジェクトを表すために使用されます。

ファイル・イベント・モニターの作成

イベント・モニターの作成時に、それが収集した情報の保管場所を決定する必要があります。ファイル・イベント・モニターは、イベント・レコードをファイルに保管します。ファイル・イベント・モニターとそのオプションは、CREATE EVENT MONITOR ステートメントによって定義されます。

ファイル・イベント・モニターを作成するには、SQLADM または DBADM 権限が必要です。

ファイル・イベント・モニターは、 8 桁の番号の付いた、拡張子 ″evt″ のファイル (例えば 00000000.evt、 00000001.evt、00000002.evt) にイベント・レコードを流します。データを小さく分割した場合でも、全体を 1 つの論理ファイルと見なす必要が

あります (つまり、データ・ストリームの開始はファイル 00000000.evt の最初のバイトであり、データ・ストリームの終了はファイル nnnnnnnn.evt 内の最後のバイトです)。イベント・モニターでは、 1 つのイベント・レコードが 2 つのファイルにわたって入ることはありません。

1. イベント・モニター・データがファイル (またはファイルのセット) に収集されることを指定し、イベント・ファイルが保管されるディレクトリーの場所を指定します。

   ```
   CREATE EVENT MONITOR dlmon FOR eventtype
                       WRITE TO FILE '/tmp/dlevents'
   ```

   dlmon はイベント・モニターの名前です。

   /tmp/dlevents は、イベント・モニターがイベント・ファイルを書き込むディレクトリー・パス (UNIX 上) の名前です。

2. モニター対象のイベントのタイプを指定します。単一イベント・モニターで、1 つ以上のイベント・タイプをモニターすることができます。

   ```
   CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                       WRITE TO FILE '/tmp/dlevents'
   ```

   このイベント・モニターは、CONNECTIONS および DEADLOCKS WITH DETAILS イベント・タイプをモニターします。

3. BUFFERSIZE 値を調整することによって、ファイル・イベント・モニター・バッファーのサイズを指定します (4K ページ単位)。

   ```
   CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                       WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
   ```

   8 は、2 つのイベント・ファイル・バッファーの容量です (4K ページ単位)。

   個々のバッファーのデフォルト・サイズは 4 ページです (16K のバッファーが 2 つ割り振られます)。最小サイズは 1 ページです。バッファーはモニター・ヒープから割り振られるので、バッファーの最大サイズはこのヒープのサイズによって制限されます。パフォーマンス上の理由で、アクティブ率の高いイベント・モニターには、アクティブ率が比較的低いものよりも大きなバッファーが必要です。

4. イベント・モニターをブロック化または非ブロック化する必要があるかどうかを指定します。ブロック化イベント・モニターの場合、イベントを生成する各エージェントは、イベント・バッファーがいっぱいであれば、それがファイルに書き込まれるまで待機します。これは、データベースのパフォーマンスを低下させることがあります。なぜなら、バッファーがクリアされるまで、延期されたエージェントおよび従属エージェントが実行できなくなるからです。イベント・データが脱落しないようにするには、BLOCKED 節を使用します。

   ```
   CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                       WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
                       BLOCKED
   ```

   イベント・モニターはデフォルトではブロック化されます。すべての単一イベント・レコードを収集することよりも、データベース・パフォーマンスのほうが重要な場合には、非ブロック化イベント・モニターを使用してください。その場合、イベントを生成する各エージェントは、イベント・バッファーがいっぱいのときに、それがファイルに書き込まれるのを待機しません。結果として、非ブロ

ック化イベント・モニターは、アクティブ率の高いシステムではデータの脱落という影響を受けます。イベント・モニターによって生じるパフォーマンス上のオーバーヘッドを最小限に抑えるには、NONBLOCKED 節を使用します。

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                    WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
                    NONBLOCKED
```

5. イベント・モニターについて収集できるイベント・ファイルの最大数を指定します。この限度に達すると、イベント・モニターは自分自身を非活動化します。

   ```
   CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                       WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
                       NONBLOCKED MAXFILES 5
   ```

   5 は、作成されるイベント・ファイルの最大数です。

   イベント・モニターが作成できるイベント・ファイルの数に制限がないことを指定することもできます。

   ```
   CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                       WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
                       NONBLOCKED MAXFILES NONE
   ```

6. イベント・モニターによって作成されるそれぞれのイベント・ファイルごとに、最大サイズを指定します (4K ページ単位)。この限度に達すると、新規ファイルが作成されます。

   ```
   CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                       WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
                       NONBLOCKED MAXFILES 5 MAXFILESIZE 32
   ```

   32 は、1 つのイベント・ファイルに入れることができる 4K ページ単位の最大数です。

   この値は、BUFFERSIZE パラメーターによって指定された値よりも大きくなければなりません。また、イベント・ファイルのサイズに制限がないことを指定することもできます。

   ```
   CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                       WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
                       NONBLOCKED MAXFILES NONE MAXFILESIZE NONE
   ```

7. データベースを開始するたびに、イベント・モニターが自動的に活動化されるかどうかを指定します。デフォルトでは、データベースの開始時にイベント・モニター (WLM イベント・モニターは例外) は自動的には活動化されません。
   - データベースの開始時に自動的に開始するイベント・モニターを作成するには、次のステートメントを発行する。

     ```
     CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                         WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
                         NONBLOCKED AUTOSTART
     ```

   - データベースの開始時に自動的に開始しないイベント・モニターを作成するには、次のステートメントを発行する。

     ```
     CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                         WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
                         NONBLOCKED MANUALSTART
     ```

8. イベント・モニターを活動化または非活動化するには、SET EVENT MONITOR STATE ステートメントを使用します。

ファイル・イベント・モニターが作成されて活動化されると、指定されたイベントが発生するたびに、モニター・データを記録します。

イベント・モニターのファイル管理

ファイル・イベント・モニターにより、イベント・モニターは、イベント・レコードをファイルに保管することができます。イベント・モニターのすべての出力は、CREATE EVENT MONITOR ステートメントの FILE パラメーターで指定されたディレクトリーに入れられます。このディレクトリーが存在しない場合、 DB2 はそれを作成しません。したがって、モニターが活動化される前に、そのディレクトリーが存在していなければなりません。そうしない場合、SET EVENT MONITOR コマンドがエラーを戻します。ファイル・イベント・モニターが最初に活動化されるとき、このディレクトリーに db2event.ctl という名前の制御ファイルが作成されます。このファイルは除去したり修正したりしないでください。

デフォルトでは、イベント・モニターはそのトレースを 00000000.evt と呼ばれる単一のファイルに書き込みます。このファイルは、ファイル・システム上にスペースがあるかぎり大きくなります。 CREATE EVENT MONITOR ステートメントの MAXFILESIZE パラメーターでファイル・サイズの限界を指定した場合には、 1 つのファイルがいっぱいになると、出力は自動的に次の番号のファイルに書き込まれます。したがって、アクティブ・ファイルは、番号の一番大きいファイルとなります。

また、CREATE EVENT MONITOR ステートメントの MAXFILES パラメーターを使って、イベント・モニター・トレース全体の最大サイズを制限できます。ファイルの数が MAXFILES で定義された最大値を超えると、イベント・モニターは自らを非活動化し、以下のメッセージが管理通知ログに書き込まれます。

```
DIA1601I Event Monitor monitor-name was deactivated when it reached
its preset MAXFILES and MAXFILESIZE limit.
```

すべてのファイルを除去することにより、この状況を避けることができます。イベント・モニターがまだ実行している間に、アクティブ・ファイルを除くすべてのイベント・ファイルを除去することができます。

ファイル・イベント・モニターがディスク・スペースを使い尽くした場合、システム・エラー・レベル・メッセージを管理通知ログに記録した後、自動的に非活動化します。

ファイル・イベント・モニターを再始動する場合、既存のデータを消去するか、既存のデータの後に新規データを追加することができます。このオプションは、CREATE EVENT MONITOR ステートメントで指定されます。そこでは、APPEND モニターまたは REPLACE モニターのどちらかを作成することができます。APPEND がデフォルトのオプションです。 APPEND イベント・モニターは、最後に使用していたファイルの終わりから書き込みを開始します。そのファイルを除去すると、次の順番のファイル番号が使用されます。追加のイベント・モニターが再始動されると、start_event だけが生成されます。イベント・ログ・ヘッダーとデータベース・ヘッダーは、最初の活動化のときに生成されます。 REPLACE イベント・モニターは常に既存のイベント・ファイルを削除し、00000000.evt から書き込みを開始します。

イベント・モニターがアクティブになっているときにモニター・データを処理する場合があります。そのことは可能です。さらに、ファイルの処理を終えたとき、そのファイルを削除し、次のモニター・データのためにスペースを解放することができます。イベント・モニターを停止して再始動しないかぎり、次のファイルに強制的に切り替えることはできません。また、APPEND モードでなければなりません。アクティブ・ファイル内でどのイベントの処理が終わったかを把握しておくために、処理された最後のレコードのファイル番号およびファイル場所だけを追跡するアプリケーションを作成することができます。次回トレースを処理するときには、アプリケーションはそのファイルの位置を検索するだけで済みます。

表書き込みイベント・モニターおよびファイル・イベント・モニターのバッファー方式

表書き込みイベント・モニターおよびファイル・イベント・モニターの場合、イベント・モニター処理はレコードをファイルまたは表に書き出す前に、それをバッファーに入れます。 バッファーがいっぱいになると、自動的にレコードの書き込みが行われます。そのため、より大きなバッファーを指定してディスク・アクセスの回数を減らせば、大量のスループットのあるイベント・モニターのモニター・パフォーマンスを改善することができます。イベント・モニターにそのバッファーをフラッシュさせるには、それを非活動化するか、または FLUSH EVENT MONITOR ステートメントを使用してバッファーを空にする必要があります。

ブロック化されたイベント・モニターは、両方のバッファーがいっぱいのときに、モニター・データを送信しているデータベース処理を一時停止します。これは、ブロック化されたイベント・モニターがアクティブのときに、イベント・レコードが廃棄されないようにするためです。一時停止されたデータベース処理とその結果として付随するデータベース処理は、バッファーが書き込まれるまでは実行できません。これは、ワークロードの種類や入出力装置の速度によっては、パフォーマンス上の大きなオーバーヘッドとなることがあります。イベント・モニターはデフォルトではブロック化されます。

ブロック化されていないイベント・モニターは、イベント・モニターがデータを書き込むことのできる速度よりも速くデータが到着するとき、エージェントから来るモニター・データを廃棄します。これにより、イベント・モニターが、他のデータベース・アクティビティーのパフォーマンスに影響しないようにします。

イベント・レコードを廃棄したイベント・モニターは、オーバーフロー・イベントを生成します。これは、モニターがイベントを廃棄した開始時刻と停止時刻、およびその期間に廃棄されたイベントの数を記述します。イベント・モニターは、ペンディングのオーバーフローがあることを報告して、終了または非活動化することがあります。その場合、次のメッセージが管理ログに書き込まれます。

```
DIA2503I Event Monitor monitor-name had a pending overflow record
when it was deactivated.
```

イベント・モニター・データの消失は、個々のイベント・レコードについても生じる可能性があります。イベント・レコードの長さがイベント・バッファリング・サイズを超えると、バッファー内に収まらないデータは切り捨てられます。例えば、stmt_text モニター・エレメントの取り込み中に、モニターされているデータベースにアタッチしたアプリケーションが長い SQL ステートメントを発行すると、この状態が生じます。イベント・レコード情報のすべてを取り込む必要がある場合に

は、より大きなバッファーを指定してください。より大きなバッファーを指定すれば、ファイルまたは表への書き込み頻度が減ることに留意してください。

パイプ・イベント・モニターの作成

イベント・モニターの作成時に、それが収集した情報の保管場所を決定する必要があります。パイプ・イベント・モニターは、イベント・モニターから Named PIPE に直接イベント・レコードを流します。

パイプ・イベント・モニターを作成するには、SQLADM または DBADM 権限が必要です。

イベント・モニターがイベント・データを書き込んですぐにデータをパイプから読み取るのは、モニター・アプリケーションの責任です。イベント・モニターがデータをパイプに書き込めない場合 (例えばパイプがいっぱいになっている場合) は、モニター・データは失われます。

パイプ・イベント・モニターは、CREATE EVENT MONITOR ステートメントで定義されます。

1. イベント・モニター・データが Named PIPE に送られることを指定します。

   ```
   CREATE EVENT MONITOR dlmon FOR eventtype
                       WRITE TO PIPE '/home/riihi/dlevents'
   ```

   dlmon はイベント・モニターの名前です。

   /home/riihi/dlevents は、イベント・モニターがイベント・レコードを送る宛先の Named PIPE (UNIX 上) の名前です。 CREATE EVENT MONITOR ステートメントは、UNIX および Windows パイプ名前構文をサポートします。

   CREATE EVENT MONITOR ステートメントで指定される Named PIPE は、イベント・モニターを活動化するときに存在し、開いている必要があります。イベント・モニターが自動的に開始するように指定する場合には、イベント・モニターの作成前に Named PIPE が存在している必要があります。
2. モニター対象のイベントのタイプを指定します。単一イベント・モニターで、1 つ以上のイベント・タイプをモニターすることができます。

   ```
   CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                       WRITE TO PIPE '/home/riihi/dlevents'
   ```

   このイベント・モニターは、CONNECTIONS および DEADLOCKS WITH DETAILS イベント・タイプをモニターします。
3. データベースを開始するたびに、イベント・モニターが自動的に活動化されるかどうかを指定します。デフォルトでは、データベースの開始時にイベント・モニターは自動的には活動化されません。
   - データベースの開始時に自動的に開始するイベント・モニターを作成するには、次のステートメントを発行する。

     ```
     CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                         WRITE TO PIPE '/home/riihi/dlevents'
                         AUTOSTART
     ```
   - データベースの開始時に自動的に開始しないイベント・モニターを作成するには、次のステートメントを発行する。

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                    WRITE TO PIPE '/home/riihi/dlevents'
                    MANUALSTART
```

4. イベント・モニターを活動化または非活動化するには、SET EVENT MONITOR STATE ステートメントを使用します。

パイプ・イベント・モニターが作成されて活動化されると、指定されたイベントが発生するたびに、モニター・データを記録します。

イベント・モニターの Named PIPE 管理

パイプ・イベント・モニターでは、Named PIPE によって、イベント・モニターのデータ・ストリームを処理することができます。パイプ・イベント・モニターの使用は、リアルタイムでイベント・レコードを処理する必要がある場合に有用です。別の利点として、アプリケーションがパイプから読み取った時に不要なデータを無視できるので、ストレージ要件を相当減らせる可能性があります。

AIX では、mkfifo コマンドを使用して Named PIPE を作成することができます。Linux および他の UNIX タイプ (Solaris オペレーティング・システムなど) では、pipe() ルーチンを使用します。 Windows では、CreateNamedPipe() ルーチンを使用して Named PIPE を作成することができます。

データをパイプに送ると、入出力は必ずブロック化され、バッファリングだけがパイプによって実行されます。イベント・モニターがイベント・データを書き込んですぐにデータをパイプから読み取るのは、モニター・アプリケーションの責任です。イベント・モニターがデータをパイプに書き込めない場合 (例えばパイプがいっぱいになっている場合) は、モニター・データは失われます。

さらに、Named PIPE には、着信するイベント・レコードを処理するための十分なスペースがなければなりません。アプリケーションが十分な速さで Named PIPE からデータを読み取らないと、パイプは満杯になり、オーバーフローが発生します。パイプ・バッファーが小さいほど、オーバーフローの発生する可能性が大きくなります。

パイプのオーバーフローが発生すると、モニターはオーバーフローが発生していることを示すオーバーフロー・イベント・レコードを作成します。イベント・モニターは OFF になりませんが、モニター・データは失われます。モニターが非活動化されるときに未解決のオーバーフロー・イベント・レコードがある場合は、診断メッセージが記録されます。それ以外の場合、パイプが書き込み可能になれば、オーバーフロー・イベント・レコードは、パイプに書き込まれます。

オペレーティング・システムでパイプ・バッファー・サイズを定義できる場合は、少なくとも 32K のパイプ・バッファーを使用してください。大ボリュームのイベント・モニターの場合、モニター・アプリケーションの処理優先順位を、エージェント処理優先順位と同等かそれ以上の値に設定する必要があります。

パーティション・データベース用のイベント・モニターの作成

パーティション・データベース・システムでファイルまたはパイプ・イベント・モニターを作成するときには、収集するモニター・データの有効範囲を決定する必要があります。

始める前に

パーティション・データベース用のイベント・モニターを作成するには、SQLADM または DBADM 権限が必要です。

このタスクについて

イベント・モニターは、オペレーティング・システムのプロセスまたはスレッドを使用して、イベント・レコードを作成します。このプロセスまたはスレッドを実行しているデータベース・パーティションのことを、モニター・パーティションといいます。ファイルおよびパイプ・イベント・モニターは、モニター・パーティション上でローカルに起こったイベントをモニターしたり、DB2 データベース・マネージャーを実行しているすべてのパーティションで起こったイベントをグローバルにモニターしたりします。グローバル・イベント・モニターは、すべてのパーティションのアクティビティーを含むトレースを、モニター・パーティション上に 1 つ作成します。イベント・モニターがローカルまたはグローバルのどちらであるかは、モニター有効範囲として示します。

モニター・パーティションおよびモニター有効範囲のどちらも、CREATE EVENT MONITOR ステートメントで指定します。

イベント・モニターは、モニター・パーティションがアクティブである場合にのみ活動化できます。イベント・モニターを活動化するために SET EVENT MONITOR ステートメントが使用されたものの、モニター・パーティションがまだアクティブではない場合には、イベント・モニターの活動化はモニター・パーティションが次回開始される時に行われます。また、イベント・モニターが明示的に非活動にされるか、インスタンスが明示的に非活動にされるまで、イベント・モニターの活動化は自動的に行われます。例えば、データベース・パーティション 0 では次のようにします。

```
db2 connect to sample
db2 create event monitor foo ... on dbpartitionnum 2
db2 set event monitor foo state 1
```

上記のコマンドを実行した後、イベント・モニター foo は、データベース sample がデータベース・パーティション 2 でアクティブになる時に自動的にアクティブになります。この自動活動化は、db2 set event monitor foo state 0 が出されるか、パーティション 2 が停止するまで行われます。

表書き込みイベント・モニターについては、ローカルまたはグローバル有効範囲の概念は該当しません。表書き込みイベント・モニターが活動化されているときには、イベント・モニターはすべてのパーティションで実行されます。 (より正確に言えば、イベント・モニター処理は、ターゲット表があるデータベース・パーティション・グループに属するパーティションに対して実行されます。) また、イベント・モニター処理が実行するそれぞれのパーティションには、同じターゲット表のセットがあります。これらの表のデータは、それがモニター・データの個々のパーティションのビューを表すという点で異なります。それぞれのパーティションのイベント・モニターのターゲット表の中の目的とする値にアクセスする SQL ステートメントを発行することによって、すべてのパーティションの集約値を入手することができます。

各ターゲット表の最初の列は PARTITION_KEY という名前で、これは表のパーティション・キーとして使用されます。この列の値は、各イベント・モニター・プロセスがそのプロセスが実行されるデータベース・パーティションにデータを挿入できるように選択されます。つまり挿入操作は、イベント・モニター・プロセスが実行されるデータベース・パーティションでローカルに実行されます。どのデータベース・パーティションでも、 PARTITION_KEY フィールドには同じ値が含まれます。このことは、データ・パーティションがドロップされてデータの再分散が行われる場合、ドロップされたデータベース・パーティション上のすべてのデータは均等に分散されるのではなく、他の 1 つのデータベース・パーティションに移動することを意味します。そのため、データベース・パーティションを除去する場合は、そのデータベース・パーティションにある表のすべての行を削除することを事前に検討してください。

その他、表ごとに PARTITION_NUMBER という名前の列を定義することができます。この列には、データが挿入されたパーティションの番号が含まれます。PARTITION_NUMBER 列は PARTITION_KEY 列と違って必須ではありません。

ターゲット表が定義されている表スペースは、イベント・モニター・データが書き込まれるすべてのパーティションに存在しなければなりません。この規則に従わないと、表スペースが存在しない (イベント・モニターがある) ログオン・パーティションにレコードが書き込まれないことになります。ただし、イベントは表スペースが存在するパーティションに書き込まれ、エラーは戻されません。この動作を利用すると、ユーザーは、特定のパーティションにのみ存在する表スペースを作成することにより、モニター用にパーティションのサブセットを選択できることになります。

表書き込みイベント・モニターの活動化の際、 FIRST_CONNECT および EVMON_START に対するコントロール表の各行は、カタログ・データベース・パーティションにのみ挿入されます。そのため、カタログ・データベース・パーティションにコントロール表のための表スペースが存在しなければなりません。そのスペースがカタログ・データベース・パーティションに存在しない場合は、これらの挿入は行われません。

表書き込みイベント・モニターが活動化されるときにパーティションがまだアクティブでない場合は、そのパーティションが次回活動化された時にイベント・モニターが活動化されます。

追加されたデータベース・パーティションがすぐにオンラインになってしまうと、イベント・モニターはその新しいパーティションにはすぐに気がつきません。 新しいパーティションに関するデータを収集したり記録するには、以下のいずれかを実行する必要があります。

- グローバル・イベント・モニターの場合は、イベント・モニターの再始動。
- 表書き込みイベント・モニターの場合は、イベント・モニターのドロップ、再作成、および再始動。

注: 詳細付きデッドロック接続イベントにおけるロック・リストには、ロックを待機しているパーティション上のアプリケーションによって保留されているロックだけが含まれます。例えば、デッドロックに関係したアプリケーションがノード 20

上でロックを待機している場合、ノード 20 上のそのアプリケーションによって保留されているロックだけがリストに含まれます。

手順

1. モニター対象のパーティションを指定します。

   ```
   CREATE EVENT MONITOR dlmon FOR DEADLOCKS
                       WRITE TO FILE '/tmp/dlevents'
                       ON PARTITION 3
   ```

   dlmon は、イベント・モニターの名前を表します。

   /tmp/dlevents は、イベント・モニターがイベント・ファイルを書き込むディレクトリー・パス (UNIX 上) の名前です。

   3 は、モニター対象のパーティション番号を表します。

2. イベント・モニター・データをローカル有効範囲で収集するか、またはグローバル有効範囲で収集するかを指定します。すべてのパーティションからのイベント・モニター・レポートを収集するには、次のステートメントを発行します。

   ```
   CREATE EVENT MONITOR dlmon FOR DEADLOCKS
                       WRITE TO FILE '/tmp/dlevents'
                       ON PARTITION 3 GLOBAL
   ```

   デッドロックおよび詳細付きデッドロック・イベント・モニターに限り、GLOBAL として定義することができます。すべてのパーティションは、デッドロック関連のイベント・レコードをパーティション 3 に報告します。

3. ローカル・パーティションからのみイベント・モニター・レポートを収集するには、次のステートメントを発行します。

   ```
   CREATE EVENT MONITOR dlmon FOR DEADLOCKS
                       WRITE TO FILE '/tmp/dlevents'
                       ON PARTITION 3 LOCAL
   ```

   これは、パーティション・データベースでのファイルおよびパイプ・イベント・モニターのデフォルトの動作です。表書き込みイベント・モニターの場合、LOCAL および GLOBAL 節は無視されます。

4. 既存のイベント・モニターのモニター・パーティションおよび有効範囲値を検討することができます。次のステートメントで、SYSCAT.EVENTMONITORS 表を照会して、このことを行います。

   ```
   SELECT EVMONNAME, NODENUM, MONSCOPE FROM SYSCAT.EVENTMONITORS
   ```

結果

イベント・モニターが作成されて活動化されると、指定されたイベントが発生するたびに、モニター・データを記録します。

イベント・モニターの出力例

コマンド行からのファイルまたはパイプ・イベント・モニター出力のフォーマット

ファイルまたはパイプ・イベント・モニターの出力はバイナリー・ストリームの論理データ・グループです。 db2evmon コマンドを使用することによって、このデー

タ・ストリームをコマンド行からフォーマットすることができます。この生産性向上ツールは、イベント・モニターのファイルまたはパイプからイベント・レコードを読み込んだ後、それらを画面に書き出します (標準出力)。

データベースに接続しているのではない限り、権限は不要です。データベースに接続している場合には、以下のいずれかが必要です。

- SYSADM
- SYSCTRL
- SYSMAINT
- DBADM

イベント・ファイルのパスを提供するか、またはデータベースおよびイベント・モニターの名前を提供することによって、どのイベント・モニターの出力をフォーマットするのかを指定することができます。イベント・モニター出力をフォーマットするには、次のようにします。

- イベント・モニター・ファイルが格納されているディレクトリーを指定する。

  ```
  db2evmon -path '/tmp/dlevents'
  ```

  `/tmp/dlevents` は (UNIX) パスです。
- データベースおよびイベント・モニター名を指定する。

  ```
  db2evmon -db 'sample' -evm 'dlmon'
  ```

  `sample` は、イベント・モニターが属するデータベースを示します。

  `dlmon` はイベント・モニターを示します。

イベント・レコードとそれに対応するアプリケーション

多数のアプリケーションがアタッチされたアクティブ・データベースのイベント・トレースでは、特定のアプリケーションに関連したイベント・レコードを追跡するのは面倒な場合があります。トレースを行いやすくするために、それぞれのイベント・レコードには、アプリケーション・ハンドルとアプリケーション ID が含まれています。これらにより、各レコードを、イベント・レコードが生成されたアプリケーションに関連付けることができます。

アプリケーション・ハンドル (**agent_id**) は、アプリケーションの使用期間中はシステム間で固有です。ただし、このハンドルは再利用されます (16 ビットのカウンターを使ってこの ID を生成します - パーティション・データベース・システムでは、この ID は、調整パーティション番号と 16 ビットのカウンターから成っています)。ほとんどの場合、この再利用は問題になりません。トレースからレコードを読み取るアプリケーションが、終了した接続を検出できるからです。例えば、 (トレースで) 既知の agent_ID を持つ接続ヘッダーを見つけたということは、この agent_ID を使っていた前の接続が終了したということです。

アプリケーション ID はタイム・スタンプを含んでいるストリング ID で、データベース・マネージャーを停止して再始動した後であっても必ず固有のままになります。

特定のアプリケーションのイベント・レコードの検出は、特に表書き込みイベント・モニターでは簡単です。イベント・モニター表では、各行がイベント・レコー

ドに対応しており、アプリケーション・ハンドルおよびアプリケーション ID が、デフォルトの列値となっています。指定されたアプリケーションのすべてのイベント・レコードを検出するには、特定のアプリケーション ID に対応するすべてのイベント・レコードについて、 SQL SELECT ステートメントを発行するだけです。

イベント・モニター自己記述型データ・ストリーム

イベント・モニターの出力はバイナリー・ストリームの論理データ・グループで、パイプ・イベント・モニターでもファイル・イベント・モニターでも全く同じです。データ・ストリームのフォーマットは、db2evmon コマンドを使用するか、またはクライアント・アプリケーションを開発することによって行えます。このデータ・ストリームは、自己記述型フォーマットで表示されます。 図 1 では、データ・ストリームの構造を示し、 89 ページの表 37 では、戻される論理データ・グループおよびモニター・エレメントのいくつかの例を示します。

注: これらの例や表では、ID として記述名を使用しています。これらの名前は、実際のデータ・ストリームでは **SQLM_ELM_** という接頭部が付きます。例えば **db_event** は、イベント・モニター出力では SQLM_ELM_DB_EVENT と表示されます。タイプは、実際のデータ・ストリームでは **SQLM_TYPE_** という接頭部が付きます。例えば、ヘッダーはデータ・ストリームで SQLM_TYPE_HEADER と表示されます。

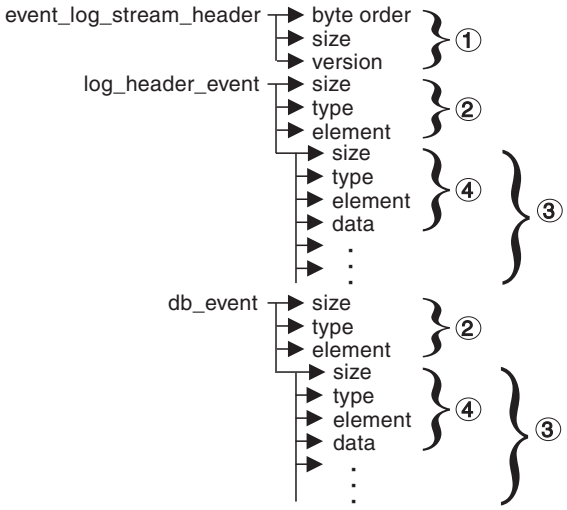


図 1. イベント・モニター・データ・ストリーム

1. sqlm_event_log_data_stream_header の構造は、データ・ストリーム内のほかのヘッダーとは異なります。バージョン・フィールドにより、出力を自己記述型データ・ストリームとして処理できるかどうかが決まります。

   このヘッダーのサイズとタイプは、バージョン 6 以前のイベント・モニター・ストリームと同じです。これにより、アプリケーションは、イベント・モニター出力が自己記述型か、バージョン 6 より前の静的形式かを判別できます。

   注: このモニター・エレメントは、データ・ストリームから sizeof(sqlm_event_log_data_stream) のバイトを読み取ることにより抽出されます。

2. 各論理データ・グループは、サイズとエレメント名を示すヘッダーで始まります。これは、 event_log_stream_header にはあてはまりません。そのサイズ・エレメントには、逆方向互換性を保持するためのダミー値が含まれるからです。
3. ヘッダーのサイズ・エレメントは、論理データ・グループのデータ全体のサイズを示します。
4. モニター・エレメント情報が、論理データ・グループ・ヘッダーに続きます。これも自己記述型です。

表 37. イベント・データ・ストリームの例

論理データ・グループ	データ・ストリーム	説明
event_log_stream_header		
	sqlm_little_endian 200 sqlm_dbmon_version9_5	使用されない (以前のリリースとの互換性のため)。 使用されない (以前のリリースとの互換性のため)。 データを戻したデータベース・マネージャーのバージョン。 イベント・モニターは自己記述型フォーマットでデータを書き込む。
log_header_event		
	100headerlog_header	論理データ・グループのサイズ。 論理データ・グループが始まることを示す。 論理データ・グループの名前。
	4u32bit byte_order little_endian	このモニター・エレメントに入っているデータのサイズ。 モニター・エレメント・タイプ - 32 ビット数値。 収集されたモニター・エレメントの名前。 このエレメントに対して収集された値。
	2u16bit codepage_id 850	このモニター・エレメントに入っているデータのサイズ。 モニター・エレメント・タイプ - 符号なし 16 ビット数値。 収集されたモニター・エレメントの名前。 このエレメントに対して収集された値。
db_event		
	100headerdb_event	論理データ・グループのサイズ。 論理データ・グループが始まることを示す。 論理データ・グループの名前。
	4u32bit lock_waits 2	このモニター・エレメントに入っているデータのサイズ。 モニター・エレメント・タイプ - 符号なし 32 ビット数値。 収集されたモニター・エレメントの名前。 このエレメントに対して収集された値。

event_log_stream_header は、データを戻したデータベース・マネージャーのバージョンを示します。イベント・モニターは自己記述型フォーマットでデータを書き込みます。スナップショット・モニターと違って、イベント・モニターにはトレースの合計サイズを戻す **size** エレメントがありません。 event_log_stream_header に示された数値は、逆方向互換性のために示されたダミー値です。イベント・トレースの

合計サイズは、 event_log_stream_header が書き込まれるときには不明です。通常は、ファイルまたはパイプの終わりに達するまで、イベント・モニター・トレースを読み取ることになります。

ログ・ヘッダーではトレースの特性を記述します。これには、トレースが収集されたサーバーのメモリー・モデル (例えばリトル・エンディアン)、およびデータベースのコード・ページなどの情報が含まれています。トレースを読み取るシステムのメモリー・モデルがサーバーとは異なる場合 (例えば、Windows 2000 システム上の UNIX サーバーからトレースを読み取る場合)、数値に関してバイトのスワッピングを行う必要があります。データベースが、トレースを読み取るマシンとは異なる言語で構成されている場合、コード・ページ変換を行う必要が生じることもあります。トレースを読み取っているとき、 **size** エレメントを使用して、トレース内の論理データ・グループを読み飛ばせます。

システム間でのイベント・モニター・データの転送

数値を保管するための規則が異なるシステム間でイベント・モニター情報を転送するときには、変換を行う必要があります。 UNIX プラットフォーム上の情報はリトル・エンディアン・バイト・オーダーで保管され、 Windows プラットフォーム上の情報はビッグ・エンディアン・バイト・オーダーで保管されます。リトル・エンディアン・ソースからのイベント・モニター・データが、ビッグ・エンディアン・プラットフォーム上で読み取られる場合、またはその逆の場合には、バイト変換が必要です。

1. 論理データ・グループ・ヘッダーおよびモニター・エレメント内の数値を変換するには、以下のロジックを使用します (C で表記)。

```
#include sqlmon.h
#define SWAP2(s) ((((s) >> 8) & 0xFF) | (((s) << 8) & 0xFF00))

#define SWAP4(l) ((((l) >> 24) & 0xFF) | ((((l) & 0xFF0000) >> 8) & 0xFF00) ¥
                      | (((l) & 0xFF00) << 8) | ((l) << 24))

#define SWAP8( where )                                                    ¥
 {                                                                        ¥
    sqluint32 temp;                                                       ¥
    temp = SWAP4(*(sqluint32 *) (where));                                 ¥
    * (sqluint32 *) (where) = SWAP4(* (((sqluint32 *) (where)) + 1));     ¥
    * (((sqluint32 *) (where)) + 1) = temp;                              ¥
 }

int HeaderByteReverse( sqlm_header_info * pHeader)
{   int rc = 0;

    pHeader->size = SWAP4(pHeader->size);
    pHeader->type = SWAP2(pHeader->type);
    pHeader->element = SWAP2(pHeader->element);

    return rc;
}

int DataByteReverse( char * dataBuf, sqluint32 dataSize)
{   int rc = 0;

    sqlm_header_info * pElemHeader = NULL;
    char * pElemData = NULL;
    sqluint32 dataOffset = 0;
    sqluint32 elemDataSize  = 0;
    sqluint32 elemHeaderSize = sizeof( sqlm_header_info);
```

```
    // For each of the elements in the data stream that are numeric,
    // perform byte reversal.

    while( dataOffset < dataSize)
    {   /* byte reverse the element header */
        pElemHeader = (sqlm_header_info *)
                      ( dataBuf + dataOffset);

        rc = HeaderByteReverse( pElemHeader);
        if( rc != 0) return rc;
        // Remember the element data's size...it will be byte reversed
        // before we skip to the next element.
        elemDataSize = pElemHeader->size;

        /* byte reverse the element data */
        pElemData = (char *)
                      ( dataBuf + dataOffset + elemHeaderSize);

        if(pElemHeader->type == SQLM_TYPE_HEADER)
        {   rc = DataByteReverse( pElemData, pElemHeader->size);
            if( rc != 0) return rc;
        }
        else
 {   switch( pElemHeader->type)
           {    case SQLM_TYPE_16BIT:
                case SQLM_TYPE_U16BIT:
                    *(sqluint16 *)(pElemData) =
                                    SWAP2(*(short *)(pElemData));
                    break;
                case SQLM_TYPE_32BIT:
                case SQLM_TYPE_U32BIT:
                    *(sqluint32 *)(pElemData) =
                                    SWAP4(*(sqluint32 *)(pElemData));
                    break;
                case SQLM_TYPE_64BIT:
                case SQLM_TYPE_U64BIT:
                    SWAP8(pElemData);
                    break;
                default:
                    // Not a numeric type. Do nothing.
                    break;
            }
        }
        dataOffset = dataOffset + elemHeaderSize + elemDataSize;
    }

    return 0;
} /*  end of DataByteReverse */
```

2. 論理データ・グループ・ヘッダーおよびモニター・エレメント内の数値を変換するには、以下のロジックを使用します (C で表記)。

```
#include sqlmon.h
#define SWAP2(s) ((((s) >> 8) & 0xFF) | (((s) << 8) & 0xFF00))

#define SWAP4(l) ((((l) >> 24) & 0xFF) | ((((l) & 0xFF0000) >> 8) & 0xFF00) ¥
                     | (((l) & 0xFF00) << 8) | ((l) << 24))

#define SWAP8( where )                                                    ¥
 {                                                                        ¥
    sqluint32 temp;                                                       ¥
    temp = SWAP4(*(sqluint32 *) (where));                                 ¥
    * (sqluint32 *) (where) = SWAP4(* (((sqluint32 *) (where)) + 1));     ¥
    * (((sqluint32 *) (where)) + 1) = temp;                              ¥
 }

int HeaderByteReverse( sqlm_header_info * pHeader)
```

```
{   int rc = 0;

    pHeader->size = SWAP4(pHeader->size);
    pHeader->type = SWAP2(pHeader->type);
    pHeader->element = SWAP2(pHeader->element);

    return rc;
}

int DataByteReverse( char * dataBuf, sqluint32 dataSize)
{   int rc = 0;

    sqlm_header_info * pElemHeader = NULL;
    char * pElemData = NULL;
    sqluint32 dataOffset = 0;
    sqluint32 elemDataSize  = 0;
    sqluint32 elemHeaderSize = sizeof( sqlm_header_info);

    // For each of the elements in the data stream that are numeric,
    // perform byte reversal.

    while( dataOffset < dataSize)
    {   /* byte reverse the element header */
        pElemHeader = (sqlm_header_info *)
                      ( dataBuf + dataOffset);

        rc = HeaderByteReverse( pElemHeader);
        if( rc != 0) return rc;
        // Remember the element data's size...it will be byte reversed
        // before we skip to the next element.
        elemDataSize = pElemHeader->size;

        /* byte reverse the element data */
        pElemData = (char *)
                      ( dataBuf + dataOffset + elemHeaderSize);

        if(pElemHeader->type == SQLM_TYPE_HEADER)
        {   rc = DataByteReverse( pElemData, pElemHeader->size);
            if( rc != 0) return rc;
        }
        else
 {   switch( pElemHeader->type)
            {    case SQLM_TYPE_16BIT:
                 case SQLM_TYPE_U16BIT:
                     *(sqluint16 *)(pElemData) =
                                     SWAP2(*(short *)(pElemData));
                     break;
                 case SQLM_TYPE_32BIT:
                 case SQLM_TYPE_U32BIT:
                     *(sqluint32 *)(pElemData) =
                                     SWAP4(*(sqluint32 *)(pElemData));
                     break;
                 case SQLM_TYPE_64BIT:
                 case SQLM_TYPE_U64BIT:
                     SWAP8(pElemData);
                     break;
                 default:
                     // Not a numeric type. Do nothing.
                     break;
             }
        }
        dataOffset = dataOffset + elemHeaderSize + elemDataSize;
    }

    return 0;
} /*  end of DataByteReverse */
```

第 4 章 スナップショット・モニター

スナップショット・モニターを使用して、データベースおよび特定の時刻に接続しているアプリケーションに関する情報をキャプチャーすることができます。スナップショットは、データベース・システムの状況を判別するのに役立ちます。一定間隔でスナップショットを取れば、傾向の監視や潜在的な問題の予測にも有用です。スナップショット・モニターの一部のデータはシステム・モニターから取得しています。 システム・モニターからどのデータが取得されるかは、システム・モニター・スイッチによって決定されます。

システム・モニターは、データベースがアクティブのときにのみ、それに関する情報を集計します。データベースからすべてのアプリケーションが切断して、データベースが非活動化すると、そのデータベースのシステム・モニター・データは入手不能になります。 ACTIVATE DATABASE コマンドを使用してデータベースを開始するか、データベースに対する永続接続を保守することにより、最後のスナップショットが取られるまでデータベースをアクティブにしておくこともできます。

スナップショット・モニターでは、インスタンス・アタッチメントが必要です。インスタンス接続がない場合、デフォルトのインスタンス接続が作成されます。インスタンス・アタッチメントは通常、初めてデータベース・システム・モニター API を呼び出す時に、DB2INSTANCE 環境変数で指定されたインスタンスに対して暗黙的に行われます。また、ATTACH TO コマンドを使用して明示的にアタッチすることもできます。一度アプリケーションがアタッチされると、そのアプリケーションが呼び出すシステム・モニター要求は、すべてアタッチ先のインスタンスにあてられます。したがって、リモートのサーバー上のインスタンスにアタッチするだけで、そのクライアントからリモート・サーバーをモニターできるようになります。

パーティション・データベース環境では、スナップショットは、インスタンスの任意のパーティションでとることも、単一のインスタンス接続を使用してグローバルにとることもできます。グローバル・スナップショットは、それぞれのパーティションで収集されたデータを集約して単一の値セットを戻します。

スナップショットは CLP または SQL 表関数からキャプチャーしたり、 C または C++ で作成されたスナップショット・モニター API を使用することによってキャプチャーすることができます。さまざまなスナップショットの要求タイプが使用可能になっており、それぞれは特定のタイプのモニター・データを戻します。例えば、バッファー・プール情報だけを戻すスナップショットや、データベース・マネージャー情報を戻すスナップショットをキャプチャーすることができます。スナップショットを取り込む前に、モニター・スイッチの制御下にあるモニター・エレメントからの情報が必要かどうかを考慮してください。あるモニター・スイッチが OFF の場合には、その制御下にあるモニター・エレメントは収集されません。

システム・モニター・データに対するアクセス権: SYSMON 権限

SYSMON データベース・マネージャー・レベルのグループに属するユーザーには、データベース・システム・モニター・データにアクセスできる権限があります。システム・モニター・データにアクセスするには、スナップショット・モニター API、CLP コマンド、または SQL 表関数を使用します。

SYSMON 権限グループは、DB2_SNAPSHOT_NOAUTH レジストリー変数に代わって、システム管理またはシステム制御権限を持たないユーザーがデータベース・システム・モニター・データにアクセスできるようにする手段になります。

スナップショット・モニターを使用してシステム・モニター・データにアクセスする方法としては、 SYSMON 権限以外には、システム管理またはシステム制御権限を持つことしかありません。

SYSMON グループに属するユーザーや、システム管理またはシステム制御権限を持つユーザーは、以下のスナップショット・モニター関数を実行できます。

- CLP コマンド:
  - GET DATABASE MANAGER MONITOR SWITCHES
  - GET MONITOR SWITCHES
  - GET SNAPSHOT
  - LIST ACTIVE DATABASES
  - LIST APPLICATIONS
  - LIST DCS APPLICATIONS
  - LIST UTILITIES
  - RESET MONITOR
  - UPDATE MONITOR SWITCHES
- API:
  - db2GetSnapshot - スナップショットの取得
  - db2GetSnapshotSize - db2GetSnapshot() 出力バッファーに必要なサイズの見積もり
  - db2MonitorSwitches - モニター・スイッチの入手/更新
  - db2ResetMonitor - モニターのリセット
- 以前に SYSPROC.SNAP_WRITE_FILE を実行していないスナップショット SQL 表関数

スナップショット管理ビューおよび表関数を使用したデータベース・システムのスナップショットのキャプチャー

許可ユーザーは、スナップショット管理ビューまたはスナップショット表関数を使用することにより、 DB2 インスタンスに関するモニター情報のスナップショットをキャプチャーできます。スナップショット管理ビューは、接続されたデータベースのすべてのデータベース・パーティションにおいてデータにアクセスするための簡単な方法を備えています。スナップショット表関数は、特定のデータベース・パーティション、グローバル集合データ、またはすべてのデータベース・パーティシ

ョンのデータに対して、データを要求できるようにします。スナップショット表関数の中は、すべてのアクティブ・データベースのデータを要求できるものもあります。

データベース・スナップショットを使用するには、SYSADM、 SYSCTRL、SYSMAINT、または SYSMON 権限が必要です。リモート・インスタンスのスナップショットを取得するには、まず、そのインスタンスに属しているローカル・データベースに接続する必要があります。

新しいモニター・データが使用できるようになった場合には将来のリリースで新しいスナップショット表関数が必要になるかもしれませんが、スナップショット管理ビューのセットはそのままで、ビューに新しい列を追加するだけです。そのため管理ビューを使用すると、アプリケーションの保守を長期間に渡って行えるという利点があります。

各スナップショット・ビューは、各データベース・パーティションのモニター対象のオブジェクトごとに 1 つの行があり、各列がモニター・エレメントを表す表を戻します。各表関数は、指定されたパーティションにおいてモニター対象のオブジェクトごとに 1 つの行を持つ表を戻します。戻される表の列名は、モニター・エレメント名と相関しています。

例えば、SAMPLE データベースに関する一般アプリケーション情報のスナップショットは、SNAPAPPL 管理ビューを使用して次のようにしてキャプチャーされます。

```
SELECT * FROM SYSIBMADM.SNAPAPPL
```

戻り表から個々のモニター・エレメントを選択することもできます。例えば、次のステートメントの場合は、**agent_id** と **appl_id** のモニター・エレメントだけが戻されます。

```
SELECT agent_id, appl_id FROM SYSIBMADM.SNAPAPPL
```

スナップショット管理ビューおよび表関数は以下のいずれかと併用できません。

- モニター・スイッチ・コマンド/API
- モニター・リセット・コマンド/API

この制約事項には、以下のコマンドが含まれます。

- GET MONITOR SWITCHES
- UPDATE MONITOR SWITCHES
- RESET MONITOR

この制限の理由は、この種のコマンドは INSTANCE ATTACH を使用するのに対して、スナップショット表関数は DATABASE CONNECT を使用するためです。

スナップショット管理ビューを使用してスナップショットをキャプチャーするには、以下のようにします。

1. スナップショット管理ビューを使用してスナップショットをキャプチャーするには、以下のようにします。
   a. データベースに接続します。これは、モニターする必要のあるインスタンス内のどのデータベースでもかまいません。スナップショット管理ビューを使用した SQL 照会は、データベースに接続していなければ発行できません。

b. キャプチャーする必要のあるスナップショットのタイプを決定します。現在接続中のデータベース以外のデータベースのスナップショットをキャプチャーする場合、または単一のデータベース・パーティションやグローバル集合データからデータを取得する場合には、代わりにスナップショット表関数を使用する必要があります。

c. 該当するスナップショット管理ビューを使用して照会を発行します。例えば、次の照会では、現在接続しているデータベースのロック情報のスナップショットをキャプチャーします。

```
SELECT * FROM SYSIBMADM.SNAPLOCK
```

2. スナップショット表関数を使用してスナップショットをキャプチャーするには、以下のようにします。

a. データベースに接続します。これは、モニターする必要のあるインスタンス内のどのデータベースでもかまいません。スナップショット表関数を使用した SQL 照会は、データベースに接続していなければ発行できません。

b. キャプチャーする必要のあるスナップショットのタイプを決定します。

c. 該当するスナップショット表関数を使用して照会を発行します。例えば、次の照会では、現在接続しているデータベース・パーティションの SAMPLE データベースに関するロック情報のスナップショットをキャプチャーします。

```
SELECT * FROM TABLE(SNAP_GET_LOCK('SAMPLE',-1)) AS SNAPLOCK
```

SQL 表関数には、以下の 2 つの入力パラメーターがあります。

データベース名

VARCHAR(255)。 NULL が入力された場合は、現在接続しているデータベースの名前が使用されます。

パーティション番号

SMALLINT。データベース・パーティション番号のパラメーターには、モニターする必要のあるデータベース・パーティションの番号に対応する整数 (0 から 999 の間の値) を入力します。現在接続しているデータベース・パーティションのスナップショットをキャプチャーする場合は、値 -1 を入力します。グローバル集合スナップショットをキャプチャーする場合は、値 -2 を入力します。すべてのデータベース・パーティションでスナップショットをキャプチャーする場合は、このパラメーターに値を指定しないでください。

注:

1) ただし、次に挙げるスナップショット表関数の場合は、現在接続しているデータベースを指定するために NULL を入力すると、インスタンス内のすべてのデータベースに関するスナップショット情報が戻されます。

- SNAP_GET_DB_V95
- SNAP_GET_DB_MEMORY_POOL
- SNAP_GET_DETAILLOG_V91
- SNAP_GET_HADR
- SNAP_GET_STORAGE_PATHS
- SNAP_GET_APPL_V95

- SNAP_GET_APPL_INFO_V95
- SNAP_GET_AGENT
- SNAP_GET_AGENT_MEMORY_POOL
- SNAP_GET_STMT
- SNAP_GET_SUBSECTION
- SNAP_GET_BP_V95
- SNAP_GET_BP_PART

2) データベース名パラメーターは、データベース・マネージャー・レベルのスナップショット表関数には適用されません。あるのは、データベース・パーティション番号用のパラメーターだけです。データベース・パーティション番号パラメーターはオプションです。

SNAP_WRITE_FILE ストアード・プロシージャーを使用した、データベース・システム・スナップショット情報のファイルへの取り込み

SNAP_WRITE_FILE ストアード・プロシージャーを使用すると、モニター・データのスナップショットを取り込み、この情報をデータベース・サーバー上のファイルに保管することができます。また、SYSADM、SYSCTRL、SYSMAINT、または SYSMON 権限を持たないユーザーがデータにアクセスすることを許可できます。これにより、すべてのユーザーがスナップショット表関数を使用した照会を行い、これらのファイルにあるスナップショット情報にアクセスできるようになります。そのため、スナップショット・モニター・データへのアクセスをオープンにすると、スナップショット表関数の実行権限を持つすべてのユーザーが、接続したユーザーのリストや、それらのユーザーがデータベースにサブミットした SQL ステートメントなどの機密情報にアクセス可能になってしまいます。スナップショット表関数の実行権限は、デフォルトで、PUBLIC に付与されています。 (ただし、表の実データやユーザー・パスワードがスナップショット・モニター表関数によって漏えいすることはありません。)

SNAP_WRITE_FILE ストアード・プロシージャーを使用してデータベース・スナップショットを取り込むには、 SYSADM、SYSCTRL、SYSMAINT、または SYSMON 権限が必要です。

SNAP_WRITE_FILE ストアード・プロシージャーへの呼び出しを発行するときは、モニターするデータベースとパーティションを識別することに加えて、*スナップショット要求タイプ* を指定する必要があります。各スナップショット要求タイプは、収集するモニター・データの有効範囲を決定します。各ユーザーが実行する必要のあるスナップショット表関数に基づいて、スナップショット要求タイプを選択してください。次の表は、スナップショット表関数とそれに対応する要求タイプのリストです。

表 38. スナップショット要求タイプ

スナップショット表関数	スナップショット要求タイプ
SNAP_GET_AGENT	APPL_ALL
SNAP_GET_AGENT_MEMORY_POOL	APPL_ALL
SNAP_GET_APPL_V95	APPL_ALL

表 38. スナップショット要求タイプ (続き)

スナップショット表関数	スナップショット要求タイプ
SNAP_GET_APPL_INFO_V95	APPL_ALL
SNAP_GET_STMT	APPL_ALL
SNAP_GET_SUBSECTION	APPL_ALL
SNAP_GET_BP_PART	BUFFERPOOLS_ALL
SNAP_GET_BP_V95	BUFFERPOOLS_ALL
SNAP_GET_DB_V95	DBASE_ALL
SNAP_GET_DETAILLOG_V91	DBASE_ALL
SNAP_GET_DB_MEMORY_POOL	DBASE_ALL
SNAP_GET_HADR	DBASE_ALL
SNAP_GET_STORAGE_PATHS	DBASE_ALL
SNAP_GET_DBM_V95	DB2
SNAP_GET_DBM_MEMORY_POOL	DB2
SNAP_GET_FCM	DB2
SNAP_GET_FCM_PART	DB2
SNAP_GET_SWITCHES	DB2
SNAP_GET_DYN_SQL_V95	DYNAMIC_SQL
SNAP_GET_LOCK	DBASE_LOCKS
SNAP_GET_LOCKWAIT	APPL_ALL
SNAP_GET_TAB_V91	DBASE_TABLES
SNAP_GET_TAB_REORG	DBASE_TABLES
SNAP_GET_TBSP_V91	DBASE_TABLESPACES
SNAP_GET_TBSP_PART_V91	DBASE_TABLESPACES
SNAP_GET_CONTAINER_V91	DBASE_TABLESPACES
SNAP_GET_TBSP_QUIESCER	DBASE_TABLESPACES
SNAP_GET_TBSP_RANGE	DBASE_TABLESPACES
SNAP_GET_UTIL	DB2
SNAP_GET_UTIL_PROGRESS	DB2

1. データベースに接続します。これは、モニターする必要のあるインスタンス内のどのデータベースでもかまいません。ストアード・プロシージャーは、データベースに接続していなければ呼び出せません。
2. スナップショット要求タイプ、およびモニターする必要のあるデータベースとパーティションを決定します。
3. スナップショット要求タイプ、データベース、およびパーティションを指定する適切なパラメーターを設定して、 SNAP_WRITE_FILE ストアード・プロシージャーを呼び出します。例えば、次の呼び出しでは、現在接続しているパーティションの SAMPLE データベースに関するアプリケーション情報のスナップショットを取り込みます。

   ```
   CALL SNAP_WRITE_FILE('APPL_ALL','SAMPLE',-1)
   ```

   SNAP_WRITE_FILE ストアード・プロシージャーには、3 つの入力パラメーターがあります。

- スナップショット要求タイプ (97 ページの表 38を参照してください。この表は、スナップショット表関数とそれに対応する要求タイプを相互参照させたものです。)
- VARCHAR (128): データベース名。 NULL が入力された場合は、現在接続しているデータベースの名前が使用されます。

  注: このパラメーターは、データベース・マネージャー・レベルのスナップショット表関数には適用されません。あるのは、要求タイプとパーティション番号のパラメーターだけです。
- SMALLINT: パーティション番号 (0 から 999 の間の値)。パーティション番号パラメーターの場合は、モニターするパーティション番号に対応する整数を入力します。現在接続しているパーティションのスナップショットを取り込む場合は、値 -1 または NULL を入力します。グローバル・スナップショットをキャプチャーするには、値 -2 を入力します。

スナップショット・データをファイルに保管した後、すべてのユーザーは、対応するスナップショット表関数を使用して照会を発行することができます。その際、データベース・レベルの表関数の入力値として (NULL, NULL) を指定し、データベース・マネージャー・レベルの表関数には (NULL) を指定します。受け取るモニター・データは、SNAP_WRITE_FILE ストアード・プロシージャーによって生成されたファイルからプルされます。

注: これにより、ユーザーによる機密モニター・データへのアクセスを制限することができますが、このアプローチにはいくつかの制限があります。

- SNAP_WRITE_FILE ファイルから入手できるスナップショット・モニター・データは、最後に SNAP_WRITE_FILE ストアード・プロシージャーが呼び出されたときと同じくらい新しいものになる。通常のインターバルで SNAP_WRITE_FILE ストアード・プロシージャーへの呼び出しを行うことによって、最新のスナップショット・モニター・データが使用可能であることを確認することができます。例えば、UNIX システム上では、これを行うために cron ジョブを設定することができます。
- スナップショット表関数を使用して照会を発行しているユーザーは、モニターするデータベースまたはパーティションを識別することができない。
  SNAP_WRITE_FILE 呼び出しを発行しているユーザーによって識別されるデータベース名およびパーティション番号が、スナップショット表関数によってアクセス可能なファイルの内容を決定します。
- ユーザーが、対応する SNAP_WRITE_FILE 要求タイプが実行されていないスナップショット表関数を含む SQL 照会を発行すると、現在接続されているデータベースおよびパーティションに対して直接スナップショットが試行されます。この操作は、ユーザーが SYSADM、SYSCTRL、SYSMAINT、または SYSMON 権限を持っている場合にのみ成功します。

SQL 照会のスナップショット表関数を使用したデータベース・システムのスナップショットへのアクセス (ファイル・アクセス使用)

許可ユーザーが SNAP_WRITE_FILE ストアード・プロシージャーを呼び出したすべての要求タイプについては、どのユーザーも、相当するスナップショット表関数を使用した照会を発行できます。ユーザーが受け取るモニター・データは、SNAP_WRITE_FILE ストアード・プロシージャーによって生成されたファイルから取り出されます。

SNAP_WRITE_FILE ファイルにアクセスすることを目的として使用されるすべてのスナップショット表関数については、許可ユーザーが、該当するスナップショット要求タイプを設定して SNAP_WRITE_FILE ストアード・プロシージャーを発行している必要があります。発行された SQL 照会に、該当する SNAP_WRITE_FILE 要求タイプが実行されていないスナップショット表関数が含まれている場合は、現在接続されているデータベースおよびパーティションへの直接のスナップショットが試行されます。この操作は、ユーザーが SYSADM、SYSCTRL、SYSMAINT、または SYSMON 権限を持っている場合にのみ成功します。

スナップショット表関数を使用して SNAP_WRITE_FILE ファイルのスナップショット・データにアクセスするユーザーには、モニターするデータベースやパーティションは識別できません。 SNAP_WRITE_FILE ファイルの内容は、ユーザーが SNAP_WRITE_FILE 呼び出しを実行して識別するデータベース名とパーティション番号によって決定されます。 SNAP_WRITE_FILE ファイルから得られるスナップショット・モニター・データは、直前に呼び出された SNAP_WRITE_FILE ストアード・プロシージャーでキャプチャーされたスナップショットだけです。

1. データベースに接続します。これは、モニターする必要のあるインスタンス内のどのデータベースでもかまいません。スナップショット表関数を使用した SQL 照会は、データベースに接続していなければ発行できません。
2. キャプチャーする必要のあるスナップショットのタイプを決定します。
3. 該当するスナップショット表関数を使用して照会を発行します。例えば、次の照会では、表スペース情報のスナップショットがキャプチャーされます。

   ```
   SELECT * FROM TABLE(SNAP_GET_TBSP_V91 (CAST(NULL AS VARCHAR(1)),
                       CAST (NULL AS INTEGER))) AS SNAP_GET_TBSP_V91
   ```

   注: データベース名やパーティション番号のパラメーターには、NULL 値を入力してください。スナップショットの対象となるデータベース名やパーティションは、 SNAP_WRITE_FILE ストアード・プロシージャーの呼び出しで決定されます。また、データベース名のパラメーターは、データベース・マネージャー・レベルのスナップショット表関数には適用されません。あるのは、パーティション番号のパラメーターだけです。
   各スナップショット表関数は、1 つ以上の行があり、各列がモニター・エレメントを表す表を戻します。したがって、各モニター・エレメントの列名は、モニター・エレメントの名前に対応しています。
4. 戻り表から個々のモニター・エレメントを選択することもできます。例えば、次のステートメントの場合は、**agent_id** のモニター・エレメントだけが戻されます。

```
SELECT agent_id FROM TABLE(
                              SNAP_GET_APPL_V95(CAST(NULL AS VARCHAR(1)),
                                                CAST (NULL AS INTEGER)))
                              as SNAP_GET_APPL_V95
```

スナップショット・モニター SQL 管理ビュー

利用可能なスナップショット・モニター SQL 管理ビューはたくさんの種類があり、それぞれがデータベース・システムの特定の領域に関するモニター・データを戻します。例えば、SYSIBMADM.SNAPBP SQL 管理ビューはバッファー・プール情報のスナップショットを取り込みます。次の表は、利用可能な各スナップショット・モニター管理ビューをリストしています。

表 39. スナップショット・モニター SQL 管理ビュー

モニター・レベル	SQL 管理ビュー	戻される情報
データベース・マネージャー	SYSIBMADM.SNAPDBM	データベース・マネージャー・レベル情報。
データベース・マネージャー	SYSIBMADM.SNAPFCM	高速コミュニケーション・マネージャー (FCM) に関するデータベース・マネージャー・レベル情報。
データベース・マネージャー	SYSIBMADM.SNAPFCM_PART	高速コミュニケーション・マネージャー (FCM) に関する、あるパーティションのデータベース・マネージャー・レベル情報。
データベース・マネージャー	SYSIBMADM.SNAPSWITCHES	データベース・マネージャーのモニター・スイッチの設定値。
データベース・マネージャー	SYSIBMADM.SNAPDBM_MEMORY_POOL	メモリー使用量についてのデータベース・マネージャー・レベル情報。
データベース	SYSIBMADM.SNAPDB	データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。
データベース	SYSIBMADM.SNAPDB_MEMORY_POOL	メモリー使用量についてのデータベース・レベル情報 (UNIX プラットフォームのみ)。
データベース	SYSIBMADM.SNAPHADR	高可用性災害時リカバリーについてのデータベース・レベル情報。
アプリケーション	SYSIBMADM.SNAPAPPL	データベースに接続されている各アプリケーションについての汎用アプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	SYSIBMADM.SNAPAPPL_INFO	データベースに接続されている各アプリケーションについての汎用アプリケーション・レベル識別情報。
アプリケーション	SYSIBMADM.SNAPLOCKWAIT	データベースに接続されているアプリケーションのロック待機についてのアプリケーション・レベル情報。

表 39. スナップショット・モニター SQL 管理ビュー (続き)

モニター・レベル	SQL 管理ビュー	戻される情報
アプリケーション	SYSIBMADM.SNAPSTMT	データベースに接続されているアプリケーションのステートメントについてのアプリケーション・レベル情報。これには、最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	SYSIBMADM.SNAPAGENT	データベースに接続されているアプリケーションに関連したエージェントについてのアプリケーション・レベル情報。
アプリケーション	SYSIBMADM.SNAPSUBSECTION	データベースに接続されているアプリケーションのアクセス・プランのサブセクションについてのアプリケーション・レベル情報。
アプリケーション	SYSIBMADM.SNAPAGENT_MEMORY_POOL	エージェント・レベルでのメモリー使用量に関する情報。
表	SYSIBMADM.SNAPTAB	データベースに接続された各アプリケーションのデータベース・レベルとアプリケーション・レベルの表アクティビティー情報。データベースに接続されたアプリケーションがアクセスした各表の表レベルの表アクティビティー情報。表スイッチが必要です。
表	SYSIBMADM.SNAPTAB_REORG	データベース内で再編成を実行している各表に関する、表レベルでの表再編成情報。
ロック	SYSIBMADM.SNAPLOCK	データベースに接続された各アプリケーションについての、データベース・レベルおよびアプリケーション・レベルのロック情報。ロック・スイッチが必要です。
表スペース	SYSIBMADM.SNAPTBSP	データベース・レベルの表スペース・アクティビティーに関する情報。また、データベースに接続された各アプリケーションのアプリケーション・レベル、およびデータベースに接続された各アプリケーションがアクセスしている各表スペースの表スペース・レベルの情報も含みます。バッファー・プール・スイッチが必要です。
表スペース	SYSIBMADM.SNAPTBSP_PART	表スペース構成に関する情報。
表スペース	SYSIBMADM.SNAPTBSP_QUIESCER	表スペース・レベルでの静止プログラムに関する情報。
表スペース	SYSIBMADM.SNAPCONTAINER	表スペース・レベルでの表スペース・コンテナー構成に関する情報。
表スペース	SYSIBMADM.SNAPTBSP_RANGE	表スペース・マップの範囲に関する情報。
バッファー・プール	SYSIBMADM.SNAPBP	指定されたデータベースのバッファー・プール・アクティビティー・カウンター。バッファー・プール・スイッチが必要です。
バッファー・プール	SYSIBMADM.SNAPBP_PART	パーティションごとに計算したバッファー・サイズおよび使用量についての情報。

表 39. スナップショット・モニター SQL 管理ビュー (続き)

モニター・レベル	SQL 管理ビュー	戻される情報
動的 SQL	SYSIBMADM.SNAPDYN_SQL	データベースの SQL ステートメント・キャッシュからのポイント・イン・タイム指定ステートメント情報。
データベース	SYSIBMADM.SNAPUTIL	ユーティリティーに関する情報。
データベース	SYSIBMADM.SNAPUTIL_PROGRESS	ユーティリティーの進行に関する情報。
データベース	SYSIBMADM.SNAPDETAILLOG	ログ・ファイルについてのデータベース・レベル情報。
データベース	SYSIBMADM.SNAPSTORAGE_PATHS	データベースの自動ストレージ・パスのリスト、および各ストレージ・パスのファイル・システム情報を戻します。

スナップショットを取り込む前に、モニター・スイッチの制御下にあるモニター・エレメントからの情報が必要かどうかを考慮してください。あるモニター・スイッチが OFF の場合には、その制御下にあるモニター・エレメントは収集されません。必要とするエレメントをスイッチで制御できるかどうか判別するには、個々のモニター・エレメントを参照してください。

すべてのスナップショット・モニター管理ビューおよび関連した表関数は、現行セッションで使用されている接続とは異なる、独立したインスタンス接続を使用します。したがって、デフォルトのデータベース・マネージャー・モニター・スイッチのみ有効です。無効なモニター・スイッチには、現行セッションまたはアプリケーションから動的に ON/OFF されるスイッチが含まれます。

DB2 バージョン 9.5 には、個々のモニター・エレメントの値を戻すだけでなく、モニター・タスクで一般的に必要とされる計算値も戻す管理ビューのセットも用意されています。例えば、SYSIBMADM.BP_HITRATIO 管理ビューはバッファー・プールのヒット率に関する計算値を戻しますが、これは複数の別個のモニター・エレメントを組み合わせたものです。

表 40. スナップショット・モニター SQL 管理用のビュー

SQL 管理用のビュー	戻される情報
SYSIBMADM.APPLICATIONS	接続されたデータベース・アプリケーションに関する情報。
SYSIBMADM.APPL_PERFORMANCE	選択された行とアプリケーションによって読み取られた行数の比率に関する情報。
SYSIBMADM.BP_HITRATIO	データベース内のバッファー・プールのヒット率 (合計、データ、および索引を含む)。
SYSIBMADM.BP_READ_IO	バッファー・プールの読み取りパフォーマンスに関する情報。
SYSIBMADM.BP_WRITE_IO	バッファー・プールの書き込みパフォーマンスに関する情報。
SYSIBMADM.CONTAINER_UTILIZATION	表スペース・コンテナーと使用率に関する情報
SYSIBMADM.LOCKS_HELD	現在保持されているロックに関する情報。
ISYSIBMADM.LOCKWAIT	ロック取得のために待機しているアプリケーションのために作動している DB2 エージェントについての情報。
SYSIBMADM.LOG_UTILIZATION	現在接続中のデータベースのログ使用率に関する情報。

表 40. スナップショット・モニター SQL 管理用のビュー (続き)

SQL 管理用のビュー	戻される情報
SYSIBMADM.LONG_RUNNING_SQL	現在接続しているデータベース内で最も長時間実行されている SQL に関する情報。
SYSIBMADM.QUERY_PREP_COST	様々な SQL ステートメントの準備に必要な時間に関する情報。
SYSIBMADM.TBSP_UTILIZATION	表スペースの構成および使用率の情報。
SYSIBMADM.TOP_DYNAMIC_SQL	実行数、平均実行時間、ソート数、またはステートメントごとのソート数を尺度にした場合に上位を占める動的 SQL ステートメント群。これらの項目に従ってソート可能。

データベース・システム・スナップショットへの SQL アクセス

スナップショット・モニター SQL 表関数 (*スナップショット表関数* と呼ばれる) を使用して、スナップショット・モニター・データにアクセスするには、以下の 2 とおりの方法があります。

- 直接アクセス
- ファイル・アクセス

直接アクセス

許可ユーザーは、スナップショット表関数で照会を発行し、モニター・データを含む結果セットを受けとることができます。この方法では、スナップショット・モニター・データへのアクセスは、 SYSADM、 SYSCTRL、 SYSMAINT、または SYSMON 権限を持つユーザーにのみ可能です。

直接アクセスを使用してスナップショット情報をキャプチャーするには、以下のようにします。

1. オプション: モニター・スイッチの状況を設定および確認します。
2. SQL を使用してデータベース・システムのスナップショットをキャプチャーします。

ファイル・アクセス

許可ユーザーは、スナップショット要求タイプ、および影響を受けるパーティションやデータベースを識別して、 SNAPSHOT_FILEW ストアード・プロシージャーを呼び出します。次に、SNAPSHOT_FILEW ストアード・プロシージャーは、データベース・サーバー上のファイルにモニター・データを保管します。

許可ユーザーが SNAPSHOT_FILEW ストアード・プロシージャーを呼び出せる各要求タイプ

これは、すべてのユーザーにスナップショット・モニター・データへのアクセスを提供する安全な方法ですが、この方法には以下の制限があります。

- SNAPSHOT_FILEW ファイルから入手できるスナップショット・モニター・データは、最後に SNAPSHOT_FILEW ストアード・プロシージャーが呼び出されたときと同じくらい新しいものになる。通常のインターバルで SNAPSHOT_FILEW ストアード・プロシージャーへの呼び出しを行うことによって、最新のスナップショット・モニター・データが使用可能で

あることを確認することができます。例えば、UNIX システム上では、これを行うために cron ジョブを設定することができます。

- スナップショット表関数を使用して照会を発行しているユーザーは、モニターするデータベースまたはパーティションを識別することができない。SNAPSHOT_FILEW 呼び出しを発行しているユーザーによって識別されるデータベース名およびパーティション番号が、スナップショット表関数によってアクセス可能なファイルの内容を決定します。
- ユーザーが、対応する SNAPSHOT_FILEW 要求タイプが実行されていないスナップショット表関数を含む SQL 照会を発行すると、現在接続されているデータベースおよびパーティションに対して直接スナップショットが試行されます。この操作は、ユーザーが SYSADM、SYSCTRL、SYSMAINT、または SYSMON 権限を持っている場合にのみ成功します。

以下のタスクは、データベース・システム・スナップショット情報をファイルにキャプチャーする SYSADM、SYSCTRL、SYSMAINT、または SYSMON ユーザーによって実行されます。

1. スナップショット要求を発行するユーザーの必要を調べます。 特に、必要なモニター・データ、収集元のデータベース、および収集が特定のパーティションに限定される必要があるかどうかを判別します。
2. オプション: モニター・スイッチの状況を設定および確認します。
3. ファイルへのデータベース・システム・スナップショット情報のキャプチャーを行います。

いったん、SYSADM、SYSCTRL、SYSMAINT、または SYSMON ユーザーが上記のステップを完了したら、すべてのユーザーは、SQL 照会内のスナップショット表関数を使用してデータベース・システム・スナップショット情報にアクセスすることができます。

CLP からのデータベース・スナップショットのキャプチャー

CLP から GET SNAPSHOT コマンドを使用して、データベース・スナップショットをキャプチャーすることができます。多数の異なるスナップショット要求タイプを使用することができます。それらには、GET SNAPSHOT コマンドに特定のパラメーターを指定することによってアクセスできます。

データベース・スナップショットを使用するには、SYSADM、 SYSCTRL、SYSMAINT、または SYSMON 権限が必要です。

データベース・スナップショットをキャプチャーするには、インスタンス・アタッチメントがなければなりません。インスタンス接続がない場合、デフォルトのインスタンス接続が作成されます。リモート・インスタンスのスナップショットを取得するには、まずそのインスタンスにアタッチする必要があります。

1. オプション: モニター・スイッチの状況を設定および確認します。
2. CLP から、GET SNAPSHOT コマンドを必要なパラメーターを指定して発行します。次の例では、データベース・マネージャー・レベル情報がスナップショットにキャプチャーされます。

```
db2 get snapshot for dbm
```

3. パーティション・データベース・システムの場合、特定のパーティションについてデータベース・スナップショットを固有にキャプチャーすることも、すべてのパーティションについてグローバルなスナップショットをキャプチャーすることもできます。特定のパーティション (例えば、パーティション番号 2) 上のすべてのアプリケーションについてのデータベース・スナップショットをキャプチャーするには、次のコマンドを発行します。

   ```
   db2 get snapshot for all applications at dbpartitionnum 2
   ```

4. すべてのパーティション上のすべてのアプリケーションについてのデータベース・スナップショットをキャプチャーするには、次のコマンドを発行します。

   ```
   db2 get snapshot for all applications global
   ```

   パーティション・データベース上のグローバル・スナップショットの場合、すべてのパーティションからのモニター・データが集約されます。

スナップショット・モニター CLP コマンド

次の表に、サポートされているスナップショット要求のタイプをすべてリストします。いくつかの要求タイプの場合、一部の情報は、関連したモニター・スイッチが ON に設定されている場合にだけ戻されます。スイッチで必須エレメントを制御できるかどうか判別するには、個々のモニター・エレメントを参照してください。

表 41. スナップショット・モニター CLP コマンド

モニター・レベル	CLP コマンド	戻される情報
接続リスト	`list applications [show detail]`	スナップショットが取られたパーティション上の DB2 インスタンスが管理するデータベースに現在接続されている、すべてのアプリケーションのアプリケーション識別情報。
接続リスト	`list applications for database dbname [show detail]`	指定のデータベースに現在接続されている各アプリケーションに関するアプリケーション識別情報。
接続リスト	`list dcs applications`	スナップショットが取られたパーティション上の DB2 インスタンスが管理するデータベースに現在接続されている、すべての DCS アプリケーションのアプリケーション識別情報。
データベース・マネージャー	`get snapshot for dbm`	インスタンス・レベルのモニター・スイッチの設定値を含む、データベース・マネージャー・レベルの情報。
データベース・マネージャー	`get dbm monitor switches`	インスタンス・レベルのモニター・スイッチの設定値。
データベース	`get snapshot for database on dbname`	データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。
データベース	`get snapshot for all databases`	パーティション上でアクティブな各データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。
データベース	`list active databases`	個々のアクティブなデータベースに対する接続数。ACTIVATE DATABASE コマンドを使用して開始したものの、接続されていないデータベースを含みます。

表 41. スナップショット・モニター CLP コマンド (続き)

モニター・レベル	CLP コマンド	戻される情報
データベース	`get snapshot for dcs database on dbname`	特定の DCS データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。
データベース	`get snapshot for remote database on dbname`	特定のフェデレーテッド・システム・データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。
データベース	`get snapshot for all remote databases`	パーティション上でアクティブな各フェデレーテッド・システム・データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。
アプリケーション	`get snapshot for application applid appl-id`	アプリケーション・レベルの情報。累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	`get snapshot for application agentid appl-handle`	アプリケーション・レベルの情報。累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	`get snapshot for applications on dbname`	パーティション上のデータベースに接続されている各アプリケーションについてのアプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	`get snapshot for all applications`	パーティション上でアクティブな各アプリケーションについてのアプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	`get snapshot for dcs application applid appl-id`	アプリケーション・レベルの情報。累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	`get snapshot for all dcs applications`	パーティション上にあるアクティブな各 DCS アプリケーションについてのアプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	`get snapshot for dcs application agentid appl-handle`	アプリケーション・レベルの情報。累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。

表 41. スナップショット・モニター CLP コマンド (続き)

モニター・レベル	CLP コマンド	戻される情報
アプリケーション	`get snapshot for dcs applications on dbname`	パーティション上にあるデータベースに接続されている各 DCS アプリケーションについてのアプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	`get snapshot for remote applications on dbname`	アプリケーション・レベルの情報。累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	`get snapshot for all remote applications`	パーティション上にあるアクティブな各フェデレーテッド・システム・アプリケーションについてのアプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
表	`get snapshot for tables on dbname`	データベースに接続された各アプリケーションのデータベース・レベルとアプリケーション・レベルの表アクティビティー情報。データベースに接続されたアプリケーションが**アクセスした**各表の表レベルの表アクティビティー情報。表スイッチが必要です。
ロック	`get snapshot for locks for application applid appl-id`	アプリケーションによって保持されているロックのリスト。ロック待機情報にはロック・スイッチが必要です。
ロック	`get snapshot for locks for application agentid appl-handle`	アプリケーションによって保持されているロックのリスト。ロック待機情報にはロック・スイッチが必要です。
ロック	`get snapshot for locks on dbname`	データベースに接続された各アプリケーションについての、データベース・レベルおよびアプリケーション・レベルのロック情報。ロック・スイッチが必要です。
表スペース	`get snapshot for tablespaces on dbname`	データベースの表スペースのアクティビティーに関する情報。バッファー・プール・スイッチが必要です。コンテナー、静止プログラム、および範囲に関する情報も含まれます。この情報はスイッチの制御下にはありません。
バッファー・プール	`get snapshot for all bufferpools`	バッファー・プール・アクティビティー・カウンター。バッファー・プール・スイッチが必要です。
バッファー・プール	`get snapshot for bufferpools on dbname`	指定されたデータベースのバッファー・プール・アクティビティー・カウンター。バッファー・プール・スイッチが必要です。
動的 SQL	`get snapshot for dynamic sql on dbname`	データベースの SQL ステートメント・キャッシュからのポイント・イン・タイム指定ステートメント情報。リモート・データ・ソースからの情報である場合もあります。

クライアント・アプリケーションからのデータベース・スナップショットのキャプチャー

C、C++、または COBOL アプリケーションでスナップショット・モニター API を使用して、データベース・スナップショットをキャプチャーすることができます。C および C++ では、db2GetSnapshot() に特定のパラメーターを指定することにより、多数の異なるスナップショット要求タイプにアクセスすることができます。

db2MonitorSwitches API を使用するには、SYSADM、SYSCTRL、SYSMAINT、または SYSMON 権限が必要です。

データベース・スナップショットをキャプチャーするには、インスタンス・アタッチメントがなければなりません。インスタンス接続がない場合、デフォルトのインスタンス接続が作成されます。リモート・インスタンスのスナップショットを取得するには、まずそのインスタンスにアタッチする必要があります。

1. オプション: モニター・スイッチの状況を設定および確認します。
2. DB2 ライブラリー、sqlmon.h および db2ApiDf.h を組み込みます。これらは sqllib の下の include サブディレクトリーにあります。

   ```
   #include <db2ApiDf.h>
   #include <sqlmon.h>
   ```

3. スナップショットのバッファー単位サイズを 100 KB に設定します。

   ```
   #define SNAPSHOT_BUFFER_UNIT_SZ 102400
   ```

4. sqlca、sqlma、db2GetSnapshotData、および sqlm_collected 構造体を宣言します。また、スナップショット・バッファーを含むようにポインターを初期化し、バッファーのサイズを設定します。

   ```
   struct sqlma *pRequestedDataGroups;
   struct sqlca sqlca;
   memset (&sqlca, '¥0', sizeof(struct sqlca));
   struct sqlm_collected collectedData;
   memset (&collectedData, '¥0', sizeof(collectedData));
   db2GetSnapshotData getSnapshotParam;
   memset (&getSnapshotParam, '¥0', sizeof(getSnapshotParam));

   static sqluint32 snapshotBufferSize = SNAPSHOT_BUFFER_UNIT_SZ;
   sqluint32 outputFormat;
   char *snapshotBuffer;
   ```

5. sqlma 構造体を初期化し、キャプチャーするスナップショットがデータベース・マネージャー・レベル情報のものであることを指定します。

   ```
   pRequestedDataGroups = (struct sqlma *)malloc(SQLMASIZE(1));
   memset(pRequestedDataGroups, '¥0', SQLMASIZE(1));
   pRequestedDataGroups->obj_num = 1;
   pRequestedDataGroups->obj_var[0].obj_type = SQLMA_DB2;
   ```

6. スナップショット出力を保持するバッファーを初期化します。

   ```
   snapshotBuffer = (char *)malloc(snapshotBufferSize);
   memset (snapshotBuffer, '¥0', snapshotBufferSize);
   ```

7. db2GetSnapshotData 構造体に、スナップショット要求タイプ (sqlma 構造体から)、バッファー情報、およびスナップショットをキャプチャーするために必要な他の情報を含めます。

   ```
   getSnapshotParam.piSqlmaData = pRequestedDataGroups;
   getSnapshotParam.poCollectedData = &collectedData;
   getSnapshotParam.poBuffer = snapshotBuffer;
   ```

```
getSnapshotParam.iVersion = SQLM_DBMON_VERSION9_5;
getSnapshotParam.iBufferSize = snapshotBufferSize;
getSnapshotParam.iStoreResult = 0;
getSnapshotParam.iNodeNumber = SQLM_CURRENT_NODE;
getSnapshotParam.poOutputFormat = &outputFormat;
getSnapshotParam.iSnapshotClass = SQLM_CLASS_DEFAULT;
```

8. スナップショットをキャプチャーします。 db2GetSnapshotData 構造体を渡します。これには、スナップショットをキャプチャーするのに必要な情報に加えて、スナップショット出力の宛先となるバッファーの参照も含まれています。

   ```
   db2GetSnapshot(db2Version810, &getSnapshotParam, &sqlca);
   ```

9. バッファーのオーバーフローを処理するためのロジックを組み込みます。スナップショットが取られた後、バッファー・オーバーフローについて sqlcode がチェックされます。バッファー・オーバーフローが発生した場合には、バッファーがクリアされて再初期化され、スナップショットが再度取られます。

   ```
   while (sqlca.sqlcode == 1606)
   {
     free(snapshotBuffer);
     snapshotBufferSize = snapshotBufferSize +
     SNAPSHOT_BUFFER_UNIT_SZ;
     snapshotBuffer = (char *)malloc(snapshotBufferSize);
     if (snapshotBuffer == NULL)
     {
       printf("¥nMemory allocation error.¥n");
       return 1;
     }
     getSnapshotParam.iBufferSize = snapshotBufferSize;
     getSnapshotParam.poBuffer = snapshotBuffer;
     db2GetSnapshot(db2Version810, &getSnapshotParam, &sqlca);
   }
   ```

10. スナップショット・モニターのデータ・ストリームを処理します。
11. バッファーをクリアします。

    ```
    free(snapshotBuffer);
    free(pRequestedDataGroups);
    ```

スナップショット・モニター API 要求タイプ

次の表に、サポートされているスナップショット要求のタイプをすべてリストします。いくつかの要求タイプの場合、一部の情報は、関連したモニター・スイッチが ON に設定されている場合にだけ戻されます。スイッチで必須エレメントを制御できるかどうか判別するには、個々のモニター・エレメントを参照してください。

表 42. スナップショット・モニター API 要求タイプ

モニター・レベル	API 要求タイプ	戻される情報
接続リスト	SQLMA_APPLINFO_ALL	スナップショットが取られたパーティション上の DB2 インスタンスが管理するデータベースに現在接続されている、すべてのアプリケーションのアプリケーション識別情報。
接続リスト	SQLMA_DBASE_APPLINFO	指定のデータベースに現在接続されている各アプリケーションに関するアプリケーション識別情報。

表 42. スナップショット・モニター API 要求タイプ (続き)

モニター・レベル	API 要求タイプ	戻される情報
接続リスト	SQLMA_DCS_APPLINFO_ALL	スナップショットが取られたパーティション上の DB2 インスタンスが管理するデータベースに現在接続されている、すべての DCS アプリケーションのアプリケーション識別情報。
データベース・マネージャー	SQLMA_DB2	インスタンス・レベルのモニター・スイッチの設定値を含む、データベース・マネージャー・レベルの情報。
データベース	SQLMA_DBASE	データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。
データベース	SQLMA_DBASE_ALL	パーティション上でアクティブな各データベースのデータベース・レベル情報およびカウンター。個々のアクティブなデータベースに対する接続数。 ACTIVATE DATABASE コマンドを使用して開始したものの、接続されていないデータベースを含みます。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。
データベース	SQLMA_DCS_DBASE	特定の DCS データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。
データベース	SQLMA_DCS_DBASE_ALL	パーティション上でアクティブな各 DCS データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。
データベース	SQLMA_DBASE_REMOTE	特定のフェデレーテッド・システム・データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。
データベース	SQLMA_DBASE_REMOTE_ALL	パーティション上でアクティブな各フェデレーテッド・システム・データベースのデータベース・レベル情報およびカウンター。情報が戻されるのは、最低 1 つのアプリケーションがデータベースに接続している場合だけです。
アプリケーション	SQLMA_APPL	アプリケーション・レベルの情報。累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	SQLMA_AGENT_ID	アプリケーション・レベルの情報。累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。

表 42. スナップショット・モニター API 要求タイプ (続き)

モニター・レベル	API 要求タイプ	戻される情報
アプリケーション	SQLMA_DBASE_APPLS	パーティション上のデータベースに接続されている各アプリケーションについてのアプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	SQLMA_APPL_ALL	パーティション上でアクティブな各アプリケーションについてのアプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	SQLMA_DCS_APPL	アプリケーション・レベルの情報。累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	SQLMA_DCS_APPL_ALL	パーティション上にあるアクティブな各 DCS アプリケーションについてのアプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	SQLMA_DCS_APPL_HANDLE	アプリケーション・レベルの情報。累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	SQLMA_DCS_DBASE_APPLS	パーティション上にあるデータベースに接続されている各 DCS アプリケーションについてのアプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	SQLMA_DBASE_APPLS_REMOTE	アプリケーション・レベルの情報。累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。
アプリケーション	SQLMA_APPL_REMOTE_ALL	パーティション上にあるアクティブな各フェデレーテッド・システム・アプリケーションについてのアプリケーション・レベル情報。これには、累積カウンター、状況情報、および最後に実行された SQL ステートメント (ステートメント・スイッチが設定されている場合) が含まれます。

表 42. スナップショット・モニター API 要求タイプ (続き)

モニター・レベル	API 要求タイプ	戻される情報
表	SQLMA_DBASE_TABLES	データベースに接続された各アプリケーションのデータベース・レベルとアプリケーション・レベルの表アクティビティー情報。データベースに接続されたアプリケーションが**アクセスした**各表の表レベルの表アクティビティー情報。表スイッチが必要です。
ロック	SQLMA_APPL_LOCKS	アプリケーションによって保持されているロックのリスト。ロック待機情報にはロック・スイッチが必要です。
ロック	SQLMA_APPL_LOCKS_AGENT_ID	アプリケーションによって保持されているロックのリスト。ロック待機情報にはロック・スイッチが必要です。
ロック	SQLMA_DBASE_LOCKS	データベースに接続された各アプリケーションについての、データベース・レベルおよびアプリケーション・レベルのロック情報。ロック・スイッチが必要です。
表スペース	SQLMA_DBASE_TABLESPACES	データベース・レベルの表スペース・アクティビティーに関する情報。また、データベースに接続された各アプリケーションのアプリケーション・レベル、およびデータベースに接続された各アプリケーションがアクセスしている各表スペースの表スペース・レベルの情報も含みます。バッファー・プール・スイッチが必要です。
バッファー・プール	SQLMA_BUFFERPOOLS_ALL	バッファー・プール・アクティビティー・カウンター。バッファー・プール・スイッチが必要です。
バッファー・プール	SQLMA_DBASE_BUFFERPOOLS	指定されたデータベースのバッファー・プール・アクティビティー・カウンター。バッファー・プール・スイッチが必要です。
動的 SQL	SQLMA_DYNAMIC_SQL	データベースの SQL ステートメント・キャッシュからのポイント・イン・タイム指定ステートメント情報。

スナップショット・モニターの出力例

スナップショット・モニターの性質を示すため、以下に、CLP を使用して取られるスナップショットの例とそれに対応する出力を示します。この例の目的は、SAMPLE データベースに接続されたアプリケーションが保持しているロックのリストを入手することです。行われるステップは次のとおりです。

1. 次のようにしてサンプル・データベースに接続します。

   ```
   db2 connect to sample
   ```

2. 次のように、UPDATE MONITOR SWITCHES コマンドを使用して「ロック」スイッチを ON にし、ロックのために待機した時間を収集します。

   ```
   db2 update monitor switches using LOCK on
   ```

3. データベース・カタログでロックを必要とするコマンドまたはステートメントを発行します。この場合、次のようにカーソルの宣言、オープン、およびフェッチを行います。

```
db2 -c- declare c1 cursor for
                        select * from staff where job='Sales' for update
db2 -c- open c1
db2 -c- fetch c1
```

4. 次のように GET SNAPSHOT コマンドを使用して、データベース・ロックのスナップショットを取ります。

```
db2 get snapshot for locks on sample
```

CLP から GET SNAPSHOT コマンドが発行された後、スナップショット出力が画面に送られます。

```
            Database Lock Snapshot

Database name                               = SAMPLE
Database path                               = C:¥DB2¥NODE0000¥SQL00001¥
Input database alias                        = SAMPLE
Locks held                                  = 5
Applications currently connected            = 1
Agents currently waiting on locks           = 0
Snapshot timestamp                          = 06-05-2002 17:08:25.048027

Application handle                          = 8
Application ID                              = *LOCAL.DB2.0098C5210749
Sequence number                             = 0001
Application name                            = db2bp.exe
CONNECT Authorization ID                    = DB2ADMIN
Application status                          = UOW Waiting
Status change time                          = Not Collected
Application code page                       = 1252
Locks held                                  = 5
Total wait time (ms)                        = 0

List Of Locks
 Lock Name                      = 0x02000300050000000000000052
 Lock Attributes                = 0x00000000
 Release Flags                  = 0x00000001
 Lock Count                     = 1
 Hold Count                     = 0
 Lock Object Name               = 5
 Object Type                    = Row
 Tablespace Name                = USERSPACE1
 Table Schema                   = DB2ADMIN
 Table Name                     = STAFF
 Mode                           = U

 Lock Name                      = 0x02000300000000000000000054
 Lock Attributes                = 0x00000000
 Release Flags                  = 0x00000001
 Lock Count                     = 1
 Hold Count                     = 0
 Lock Object Name               = 3
 Object Type                    = Table
 Tablespace Name                = USERSPACE1
 Table Schema                   = DB2ADMIN
 Table Name                     = STAFF
 Mode                           = IX

 Lock Name                      = 0x0100000001000000100810056
 Lock Attributes                = 0x00000000
 Release Flags                  = 0x40000000
 Lock Count                     = 1
```

```
Hold Count                  = 0
Lock Object Name            = 0
Object Type                 = Internal Variation Lock
Mode                        = S

Lock Name                   = 0x41414141414A48520000000041
Lock Attributes             = 0x00000000
Release Flags               = 0x40000000
Lock Count                  = 1
Hold Count                  = 0
Lock Object Name            = 0
Object Type                 = Internal Plan Lock
Mode                        = S

Lock Name                   = 0x434F4E544F4B4E310000000041
Lock Attributes             = 0x00000000
Release Flags               = 0x40000000
Lock Count                  = 1
Hold Count                  = 0
Lock Object Name            = 0
Object Type                 = Internal Plan Lock
Mode                        = S
```

このスナップショットを見ると、現在 1 つのアプリケーションが SAMPLE データベースに接続しており、 5 つのロックを保持していることが分かります。

```
Locks held                                = 5
Applications currently connected          = 1
```

Application status が UOW Waiting になった時刻 (Status change time) は Not Collected として戻されることに注意してください。これは、「作業単位」スイッチが OFF であるためです。

ロック・スナップショットは、このデータベースに接続しているアプリケーションがロックのために待機している、現在までの合計時間も戻します。

```
Total wait time (ms)                      = 0
```

サブセクション・スナップショット

パーティション間並列処理を使用しているシステムでは、SQL コンパイラーによって、 SQL ステートメントのアクセス・プランがサブセクションに区分化されます。個々のサブセクションは別々の DB2 エージェント (または SMP のエージェント) によって実行されます。

コンパイル時に DB2 コード生成プログラムによって生成される SQL ステートメントのアクセス・プランを取得するには、 db2expln コマンドを使用します。一例として、複数のパーティションにパーティション化されている表の行をすべて選択すると、アクセス・プランは以下の 2 つのサブセクションに分けられます。

1. サブセクション 0。コーディネーター・サブセクション。この役割は、他の DB2 エージェント (サブエージェント) にフェッチされた行を収集してアプリケーションに戻すことです。
2. サブセクション 1。この役割は、表スキャンを実行して、行をコーディネーター・エージェントに戻すことです。

この単純な例では、サブセクション 1 はすべてのデータベース・パーティションに分散されます。この表が属するデータベース・パーティション・グループの個々の物理パーティションに、このサブセクションを実行するサブエージェントがあります。

データベース・システム・モニターを使用すると、ランタイムの情報とアクセス・プラン (コンパイル時の情報) とを関連付けることができます。パーティション間並列処理により、モニターは情報をサブセクションのレベルに分類します。例えば、ステートメント・モニターのスイッチが ON になっている場合に、 GET SNAPSHOT FOR APPLICATION を実行すると、ステートメント全体の情報と、このパーティション上で実行される個々のサブセクションに関する情報が戻されます。

アプリケーションのスナップショットを実行すると、以下のサブセクション情報が戻されます。

- 読み書きされた表の行数
- CPU の使用量
- 経過時間
- このステートメントで作業する他のエージェントとの間で送受信される表キューの行数。これにより、スナップショットを連続してとることで、実行時間の長い照会の実行状況を追跡できます。
- サブセクションの状況。サブセクションが WAIT 状態 (別のエージェントがデータを送受信するのを待機している) の場合は、この情報によって、サブセクションの実行を妨げているパーティションも識別できます。識別した後にそのパーティションでスナップショットを取って状態を調べることができます。

この情報は、ステートメント・イベント・モニターによって、サブセクションごとに実行完了後に記録されます。この情報には CPU 使用量、合計実行時間、および他の複数のカウンターが含まれます。

パーティション・データベース・システムでのグローバル・スナップショット

パーティション・データベース・システムでは、スナップショット・モニターを使用して現行パーティション、指定したパーティション、またはすべてのパーティションのスナップショットをとることができます。パーティション・データベースのすべてのパーティションに渡ってグローバル・スナップショットをとる場合は、データが集約されてから、結果が戻されます。

データは、以下のようなさまざまなエレメント・タイプについて集約されます。

- **カウンター、時間、ゲージ**

  インスタンス内のそれぞれのパーティションから収集されたすべての同種値の合計が入っています。例えば、GET SNAPSHOT FOR DATABASE XYZ ON TEST GLOBAL は、パーティション・データベース・インスタンス内のすべてのパーティションについて、データベースから読み取られた行数 (rows_read) を戻します。
- **水準点**

パーティション・データベース・システム内のパーティションについて検出される最高値 (高位水準点) または最低値 (低位水準点) を戻します。戻された値に注目する場合は、個々のパーティションのスナップショットを取って、特定のパーティションが使用過剰になっているのか、またはインスタンス全体に問題があるのかを判別することができます。

- **タイム・スタンプ**

  スナップショット・モニター・エージェントがアタッチされているパーティションのタイム・スタンプ値に設定します。すべてのタイム・スタンプ値は、「タイム・スタンプ」モニター・スイッチの制御下にあります。

- **情報**

  作業を妨害している可能性のあるパーティションに関する最も重要な情報を戻します。例えば、エレメント `appl_status` について、あるパーティションでの状況が「UOW 実行」であり、別のパーティションでは「ロック待機」の場合には、「ロック待機」が戻されます。なぜなら、これはアプリケーションの実行を保留にしている状況だからです。

さらに、カウンターのリセット、モニター・スイッチの設定、モニター・スイッチ設定値の検索はパーティション・データベース内の個々のパーティション、またはすべてのパーティションに対して行うことができます。

注: グローバル・スナップショットをとる際に、1 つ以上のパーティションでエラーが生じると、データはスナップショットを正常にとることのできたパーティションから収集され、さらに警告 (sqlcode 1629) が戻されます。モニター・スイッチのグローバル取得または更新が失敗したり、 1 つ以上のパーティションでカウンター・リセットが失敗すると、それらのパーティションではスイッチの設定やデータのリセットは行われません。

スナップショット・モニター自己記述型データ・ストリーム

db2GetSnapshot API でスナップショットをキャプチャーした後、スナップショット出力が自己記述型データ・ストリームとして戻されます。 118 ページの図 2 では、データ・ストリームの構造を示し、 118 ページの表 43 では、戻される論理データ・グループおよびモニター・エレメントのいくつかの例を示します。

注: これらの例や表では、ID として記述名を使用しています。これらの名前は、実際のデータ・ストリームでは **SQLM_ELM_** という接頭部が付きます。例えば collected は、スナップショット・モニター出力で SQLM_ELM_COLLECTED と表示されます。タイプは、実際のデータ・ストリームでは **SQLM_TYPE_** という接頭部が付きます。例えば、ヘッダーはデータ・ストリームで SQLM_TYPE_HEADER と表示されます。

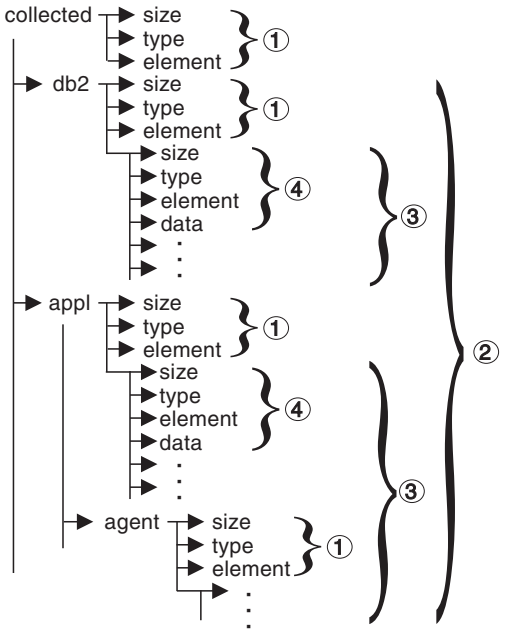


図 2. スナップショット・モニター・データ・ストリーム

1. 各論理データ・グループは、サイズと名前を示すヘッダーで始まります。このサイズには、ヘッダー自体に取られるデータ・ボリュームは組み込まれません。
2. collected ヘッダー内のサイズは、スナップショットの合計サイズを戻します。
3. ほかのヘッダー内のサイズ・エレメントは、その論理データ・グループのデータ全体のサイズを示します (従属のグループをすべて含む)。
4. モニター・エレメント情報が、論理データ・グループ・ヘッダーに続きます。これも自己記述型です。

表 43. スナップショット・データ・ストリームの例

論理データ・グループ	データ・ストリーム	説明
collected		
	1000	スナップショット・データのサイズ (バイト)。
	headercollected	論理データ・グループが始まることを示す。
		論理データ・グループの名前。
	4u32bit	このモニター・エレメントに入っているデータのサイズ。
	server_db2_type	モニター・エレメント・タイプ - 符号なし 32 ビット数値。
	sqlf_nt_server	収集されたモニター・エレメントの名前。
		このエレメントに対して収集された値。
	2u16bit	このモニター・エレメントに入っているデータのサイズ。
	node_number	モニター・エレメント・タイプ - 符号なし 16 ビット数値。
	3	収集されたモニター・エレメントの名前。
		このエレメントに対して収集された値。

表 43. スナップショット・データ・ストリームの例 (続き)

論理データ・グループ	データ・ストリーム	説明
db2		
	200 headerdb2	スナップショットのデータの DB2 レベル部分のサイズ。 論理データ・ グループが始まることを示す。 論理データ・グループの名前。
	4u32bit sort_heap_allocated 16	このモニター・エレメントに入っているデータのサイズ。 モニター・エレメント・タイプ - 符号なし 32 ビット数値。 収集されたモニター・エレメントの名前。 このエレメントに対して収集された値。
	4u32bit local_cons3	このモニター・エレメントに入っているデータのサイズ。 モニター・エレメント・タイプ - 符号なし 32 ビット数値。 収集されたモニター・エレメントの名前。 このエレメントに対して収集された値。
	. . .	. . .
appl		
	100headerappl	スナップショットの appl エレメント・ データのサイズ。 論理データ・グルー プが始まることを示す。 論理データ・グループの名前。
	4u32bit locks_held 3	このモニター・エレメントに入っているデータのサイズ。 モニター・エレメント・タイプ - 符号なし 32 ビット数値。 収集されたモニター・エレメントの名前。 このエレメントに対して収集された値。
	. . .	. . .
agent		
	50 headeragent	appl 構造のエージェント部分のサイズ。 論理データ・グループが始まることを示す。 論理データ・グループの名前。
	4u32bit agent_pid 12	このモニター・エレメントに入っているデータのサイズ。 モニター・エレメント・タイプ - 32 ビット数値。 収集されたモニター・エレメントの名前。 このエレメントに対して収集された値。
	. . .	. . .

db2GetSnapshot() ルーチンは、自己記述型スナップショット・データを、ユーザー提供のバッファーに戻します。データは、キャプチャーされたスナップショットのタイプに関連する論理データ・グループに分けられて戻されます。

スナップショット要求によって戻される各アイテムには、そのサイズおよびタイプを指定するフィールドが含まれます。サイズを使用して、戻りデータ全体を解析できます。また、フィールド・サイズを使用して、論理データ・グループを読み飛ばすこともできます。例えば、DB2 レコードを読み飛ばすには、データ・ストリーム内のバイト数を判別する必要があります。読み飛ばすバイト数を計算するには、次の公式を使用してください。

```
db2 論理データ・グループのサイズ + sizeof(sqlm_header_info)
```

対話モードで db2top を実行してモニターするときに使用できるコマンド

db2top モニター・ユーティリティーは、複雑な DB2 環境を短時間で効率的にモニターするためのツールです。すべてのデータベース・パーティションの DB2 スナップショット情報を結合して、実行中の DB2 システムの動的なリアルタイム・ビューを提供できます。このツールの操作には、テキスト・ベースのユーザー・インターフェースを使用します。

対話モードで db2top を実行する場合は、以下のコマンドを実行できます。

A HADR クラスターの 1 次データベースと 2 次データベースのいずれかをモニターします。

a エージェントのアプリケーション詳細に移動します (ステートメント画面ではエージェントに関する制限に移動します)。db2top コマンドから、エージェント ID を入力するためのプロンプトが表示されます。

B 重要なサーバー・リソースの主なコンシューマーを表示します (ボトルネック分析)

c このオプションによって、画面に表示する列の順序を変更できます。構文は 1,2,3,... という形式です (1,2,3 はそれぞれ、表示する 1 番目、2 番目、3 番目の列に相当します)。これらの数字は、ソート基準を指定するときに使用する列番号にもなります。

c スイッチ・キーを使用すると、画面に表示する列の順序を指定するための画面が表示されます。画面の左側にはデフォルトの順序と列番号が表示され、画面の右側には現在の配列が表示されます。列の順序を変更するには、画面の下部にあるテキスト・フィールドに新しい列の順序を入力します。次に、左側に表示されている列の相対的な位置を入力し、それぞれをコンマで区切ります。すべての列を指定しなければならないわけではありません。w を選択すれば、この列の配列を *$DB2TOPRC* に保存して、後続の db2top モニター・セッションで使用できるようになります。画面に表示する列をソートして、どの順序で配列するかを選択できます。 `.db2toprc` ファイルの中で列の配列のために使用できる有効なキーワードは、以下のとおりです。

- sessions=
- tables=
- tablespaces=
- bufferpools=

- dynsql=
- statements=
- locks=
- utilities=
- federation=

b バッファー・プール画面に移動します。

C スナップショット・データ収集プログラムのオン/オフを切り替えます。

d データベース画面に移動します。

D 動的 SQL 画面に移動します。

f 画面をフリーズします。

F 1 次サーバーでフェデレーテッド照会をモニターします。

G グラフのオン/オフを切り替えます。

h ヘルプ画面に移動します。

H 履歴画面に移動します。

i アイドル・セッションのオン/オフを切り替えます。

k 実際の値と差分の値を切り替えます。

l セッション画面に移動します。

L SQL 画面から完全な照会テキストを表示します。その後、e オプションまたは X オプションを使用して、通常の DB2 Explain を実行できます。

m メモリー・プールを表示します。

o セッション・セットアップを表示します。

p パーティション画面に移動します。

P スナップショットを実行するデータベース・パーティションを選択します。

q db2top を終了します。

R スナップショット・データをリセットします。

s ステートメント画面に移動します。

S ネイティブ DB2 スナップショットを実行します。

t 表スペース画面に移動します。

T 表画面に移動します。

u アクティブなユーティリティーを表示して、各データベース・パーティションの情報を集約します。

U ロック画面に移動します。

V デフォルトの Explain スキーマを設定します。

w セッション設定を .db2toprc に書き込みます。

W agent_id、os_user、db_user、application、netname の監視モード。セッション・スナップショットから返されるステートメント (オプション l) は、agent.sql、os_user-agent.sql、db_user-agent.sql、application-agent.sql、netname-

agent.sql に書き込まれます。動的 SQL 画面から実行すると (オプション D)、ステートメントは、db2advis と互換性のある形式で db2adv.sql に書き込まれます。

X 拡張モードのオン/オフを切り替えます。

z|Z 昇順または降順でソートします。

/ データのフィルターとして使用する式を入力します。式は正規表現に準拠している必要があります。機能 (画面) ごとに別々のフィルターを適用できます。行全体に regexp チェックが適用されます。

<|> 画面の左側または右側に移動します。

以下のスイッチは、アプリケーション画面だけに該当します。

r 前の機能に戻ります。

R 自動リフレッシュを切り替えます。

g グラフのオン/オフを切り替えます。

X 拡張モードのオン/オフを切り替えます。

d エージェントを表示します。

対話モードで db2top を開始する場合は、以下のコマンドを実行します。

```
db2top -d <database name>
```

以下のコマンドを入力します。

```
db2top -d sample
```

以下の出力が表示されます。

```
[\]11:57:10,refresh=2secs(0.000) Inactive,part=[1/1],<instanceName>:sample
[d=Y,a=N,e=N,p=ALL] [qp=off]

[/]: When rotating, it means that db2top is waiting between two snapshots, otherwise, it means db2top is waiting
     from an answer from DB2
11:57:10: current time
refresh=2secs: time interval
refresh=!secs: Exclamation mark means the time to process the snapshot by DB2 is longer than the refresh interval.
     In this case, db2top will increase the interval by 50%. If this occurs too often because the system is too busy,
     you can either increase the snapshot interval (option I), monitor a single database partition (option P), or turn
     off extended display mode (option x)
0.000 : time spent inside DB2 to process the snapshot
d=Y/N : delta or cumulative snapshot indicator (command option -k or option k).
a=Y/N : Active only or all objects indicator (-a command option set or i)
e=Y/N : Extended display indicator
p=ALL : All database partitions
p=CUR: Current database partition (-P command option with no partition number specified)
p=3 : target database partition number: say 3

Inactive: : Shows inactive if DB2 is not running, otherwise displays the platform on which DB2 is running
part=[1/1] : active database partition number vs total database partition number. For example, part=[2,3] means one
      database partition out of 3 is down (2 active, 3 total)
<instanceName> : instance name
sample : database name
qp=off/on : query patroller indicator (DYNMGMT database configuration parameter) for the database partition
      on which db2top is attached
```

パーティション・データベース環境で db2top モニター・ユーティリティーを対話モードで実行する場合の例を以下に示します。

```
db2top -d TEST -n mynode -u user -p passwd -V skm4 -B -i 1
The command parameters are as follows:
-d TEST     # database name
```

```
-n mynode   #  node name
-u user     #  user id
-p passwd   #  password
-V skm4     #  Schema name
-B          #  Bold enabled
-i 1        #  Screen update interval: 1 second
```

.db2toprc 構成ファイル

.db2toprc 構成ファイルは、db2top モニター・ユーティリティーの初期化時にパラメーターを設定するために使用するユーザー生成ファイルです。db2top ユーティリティーは、ユーザー定義変数 *$db2topRC* を使用して、.db2toprc ファイルの場所を検索します。その変数が設定されていない場合、db2top はまず現行ディレクトリーで .db2toprc ファイルを検索してから、home ディレクトリーの中を検索します。.db2toprc は、ユーザー生成ファイルです。

環境変数

以下の環境変数を設定できます。

- DB2TOPRC

  .db2toprc ファイルの場所を格納するユーザー定義環境変数。例えば、Linux の場合は、DB2TOPRC を `export db2topRC="~/db2top"` と定義できます。

  その変数がユーザーによって設定されていない場合、db2top はまず現行ディレクトリーで .db2toprc ファイルを検索してから、`home` ディレクトリーの中を検索します。

- DB2DBDFT

  この変数では、暗黙接続で使用するデータベースのデータベース別名を指定します。コマンド行または .db2toprc 構成ファイルでデータベース名が指定されていない場合に、その変数が使用されます。

- EDITOR

  このシステム環境変数では、Explain スナップショットまたはネイティブ・スナップショットの結果を表示するためのテキスト・エディターを開始するときに使用するコマンドを指定します。

  この変数が設定されていない場合は、vi が使用されます。

構造

ここでは、.db2toprc ファイルのいくつかの項目を取り上げます。

cpu=command

この項目を使用して、画面出力の右側の第 2 行に CPU アクティビティーの結果を表示します。以下に例を示します。

`cpu=vmstat 2 2 | tail -1 | awk '{printf("%d(usr+sys)0,$14+$15);}'` 画面の右側に `Cpu=2(usr+sys)` が表示されます。

io=command

この項目を使用して、コマンドを指定し、画面出力の左側の第 2 行に結果を表示します。以下に例を示します。

`io=vmstat 2 2 | tail -1 | awk '{printf("%d(bi+bo)0,$10+$11);}'` 画面の左側に `Disk=76(bi+bo)` が表示されます。

どちらのコマンドもバックグラウンド・プロセスとして実行され、画面の各フィールドが非同期モードで更新されます。

shell alias=command

このシェル項目を使用して、ユーザー定義コマンドを指定します。例えば、`shell M=top` を指定した場合は、`M` を入力すると、db2top セッションからトップが作成されます。終了時には、現在の画面に戻ります。

function alias=command

この項目を使用して、ユーザー定義コマンドを指定します。例えば、`function N=netstat` を指定した場合は、netstat の出力を繰り返し表示する N という新しい関数が作成されます。`function` 項目は、複数記述できます。それぞれの項目を別々の行に配置する必要があります。以下に例を示します。

```
function Q=netstat
function N=df -k
```

sort=command

この項目を使用して、ソート順を指定します。例えば、`sort=command` を指定すると、その関数のデフォルトのソート順が作成されます (command は列の番号です)。昇順と降順のいずれかになります。ソートは、sessions、tables、tablespace、bufferpool、dynsql、statements、locks、utilities、federation で有効です。

サンプル .db2toprc ファイル

デフォルトの .db2toprc 構成ファイルはありません。ただし、「W」を押せば、現在のセットアップの .db2toprc を作成できます。以下のサンプル .db2toprc ファイルを参考にしてください。すべての項目にコメントを追加しています。

```
# db2top configuration file
# On UNIX, should be located in $HOME/<cmdname> .db2toprc</cmdname>
# File generated by db2top-1.0a
#
node= # [-n] nodename
database=sample # [-d] databasename
user= # [-u] database user
password= # [-p] user password (crypted)
schema= # [-V] default schema for explains
interval=2 # [-i] sampling interval
active=OFF # [-a] display active sessions only (on/off)
reset=OFF # [-R] Reset snapshot at startup (on/off)
delta=ON # [-k] Toggle display of delta/cumulative values (on/off)
gauge=ON # display graph on sessions list (on/off)
colors=ON  # True if terminal supports colors. Informs GE_WRS if it can display information with colors
graphic=ON  # True if terminal supports semi graphical characters (on/off).
port= # Port for network collection
streamsize=size # Max collection size per hour (eg. 1024 or 1K : K, M or G)
# Command to get cpu usage information from OS
cpu=vmstat 2 2 | tail -1 | awk '{printf("%d(usr+sys)0,$14+$15);}'
# Command to get IO usage information from OS
io=vmstat 2 2 | tail -1 | awk '{printf("%d(bi+bo)0,$10+$11);}'
# Ordering of information in sessions screen
# Column order for the session screen (option l)
sessions=0,1,18,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,19,20,21,22,23
```

```
# Column order for the tables screen (option T)
tables=0,1,2,4,3,5,6,7
# Column order for the tablespaces screen (option t). The display will be sorted in ascending order on column #22
tablespaces=0,1,18,2,3,4,5,6,7,8, sort=22a
# Column order for the bufferpool screen (option b)
bufferpools=0,1,18,2,3,4,5,6,7,8,9,10
# Column order for the Dynamic SQL screen (option D)
dynsql=0,1,18,2,3,4,5,6,7,8,9
statements=0,1
locks=0,1
utilities=0 # contains the default column and sort order for the utility screen
federation=0,2,4  # contains the default column and sort order for the federation screen

# User defined commands
shell P=top
function N=date && netstat -t tcp
```

第 5 章 スイッチ・ベースのモニターの概念

システム・モニター・スイッチ

システム・モニター・スイッチはスナップショット・モニターや一部のイベント・モニターがデータを収集する方法を制御します。

注: これらのシステム・モニター・スイッチによる、作業単位イベント・モニターおよびロック・イベント・モニター (DB2 バージョン 9.7 で導入されています) への影響はありません。

スナップショット・モニターおよび一部のイベント・モニターは、システム・モニターが収集したデータを報告します。 システム・モニター・データを収集すると、データベース・マネージャーの処理オーバーヘッドが発生します。例えば、SQL ステートメントの実行時間を計算するには、すべてのステートメントの実行前後にデータベース・マネージャーがオペレーティング・システムを呼び出して、タイム・スタンプを入手しなければなりません。この種のシステム呼び出しには通常はコストがかかります。システム・モニターによって発生するオーバーヘッドの別の形は、メモリー使用量の増加です。システム・モニターによって追跡されるすべてのモニター・エレメントについて、データベース・マネージャーはメモリーを使用し、収集されたデータを保管します。

モニター情報の保守に関連したオーバーヘッドを最小限にとどめるために、モニター・スイッチを使って、高コストになりかねないデータベース・マネージャーによるデータ収集を制御します。各スイッチには、ON と OFF という 2 つの設定値だけがあります。モニター・スイッチが OFF の場合には、そのスイッチの制御下にあるモニター・エレメントは情報を収集しません。スイッチの制御下にはなく、またスイッチの設定に関係なく常に収集される、基本モニター・データは相当量あります。

それぞれのモニター・アプリケーションには、モニター・スイッチ (およびシステム・モニター・データ) の独自の論理ビューがあります。始動時に、各アプリケーションはそのモニター・スイッチ設定値を、(インスタンス・レベルの) データベース・マネージャー構成ファイル内の dft_monswitches パラメーターから継承します。モニター・アプリケーションは UPDATE MONITOR SWITCHES USING MONSWITCH OFF/ON コマンドを使用して、モニター・スイッチ設定値を変更することができます。 MONSWITCH パラメーターは、下記の「スナップショット・モニター・スイッチ」表の「モニター・スイッチ」欄の値を保持しています。スイッチの設定値をアプリケーション・レベルで変更すると、スイッチを変更したアプリケーションだけに影響が及びます。

インスタンス・レベルのモニター・スイッチの変更は、データベース管理システムを停止せずに行うことができます。これを行うには、UPDATE DBM CFG USING DBMSWITCH OFF/ON コマンドを使用します。 DBMSWITCH パラメーターは、下記の「スナップショット・モニター・スイッチ」表の「DBM パラメーター」欄の値を保持しています。このようにスイッチを動的に更新する際、更新を動的に有効にするためには、更新を実行しているアプリケーションを明示的にインスタンスにアタ

ッチする必要があります。動的な更新を行っても、他の既存のスナップショット・アプリケーションへの影響はありません。新規のモニター・アプリケーションは、更新済みのインスタンス・レベルのモニター・スイッチ設定値を継承します。既存のモニター・アプリケーションが新規のデフォルトのモニター・スイッチ値を継承するには、いったん終了して、そのアタッチメントを再構築する必要があります。データベース・マネージャー構成ファイルのスイッチを更新すると、パーティション・データベース内のすべてのパーティションのスイッチも更新されます。

データベース・マネージャーでは、スナップショット・モニター・アプリケーションとそのスイッチの設定値がすべて追跡されます。 1 つのアプリケーションの構成内でスイッチが ON に設定されている場合は、データベース・マネージャーは必ずモニター・データを収集します。したがって、アプリケーションの構成内で、あるスイッチが OFF になっている場合でも、その同じスイッチが ON になっているアプリケーションが少なくとも 1 つある限り、データベース・マネージャーはデータを収集します。

時間およびタイム・スタンプ・エレメントの収集は、「タイム・スタンプ」スイッチによって制御されます。このスイッチを OFF にすると (デフォルトでは ON)、時間またはタイム・スタンプに関連したモニター・エレメントを判別するときにオペレーティング・システムが呼び出すタイム・スタンプをすべてスキップするよう、データベース・マネージャーに指示されます。 CPU の使用率が 100% に近づいたときには、このスイッチを OFF にすることが重要となります。そのような状況では、タイム・スタンプの発行によって、かなりのパフォーマンス低下が生じます。「タイム・スタンプ」スイッチと他のスイッチによって制御できるモニター・エレメントの場合は、それらのスイッチのいずれかを OFF にするとデータは収集されません。そのため、「タイム・スタンプ」スイッチを OFF にすると、他のモニター・スイッチの制御下にあるデータの全体コストが大幅に減少します。

イベント・モニターは、スナップショット・モニター・アプリケーションと同じような、モニター・スイッチによる影響は受けません。イベント・モニターが定義されると、指定されたイベント・タイプにより必要とされるインスタンス・レベルのモニター・スイッチを、自動的に ON にします。例えば、デッドロック・イベント・モニターは、「ロック」モニター・スイッチを自動的に ON にします。イベント・モニターが活動化されると、必要なモニター・スイッチが ON になります。イベント・モニターが非活動化されると、モニター・スイッチは OFF になります。

「タイム・スタンプ」モニター・スイッチは、イベント・モニターによって自動的に設定されません。これは、イベント・モニターの論理データ・グループに属するモニター・エレメントの収集を制御する、唯一のモニター・スイッチです。「タイム・スタンプ」スイッチが OFF の場合は、イベント・モニターによって収集されるタイム・スタンプおよび時間モニター・エレメントの大半が収集されません。これらのエレメントは、依然として指定された表やファイル、またはパイプに書き込まれますが、その値はゼロとなります。

表 44. スナップショット・モニター・スイッチ

モニター・スイッチ	DBM パラメーター	提供される情報
BUFFERPOOL	DFT_MON_BUFPOOL	読み取りおよび書き込みの数、かかった時間

表 44. スナップショット・モニター・スイッチ (続き)

モニター・スイッチ	DBM パラメーター	提供される情報
LOCK	DFT_MON_LOCK	ロック待機時間、デッドロック
SORT	DFT_MON_SORT	使用されたヒープ数、ソート・パフォーマンス
STATEMENT	DFT_MON_STMT	開始/停止時刻、ステートメント識別
TABLE	DFT_MON_TABLE	アクティビティーの程度 (読み取られた/書き込まれた行)
UOW	DFT_MON_UOW	開始/終了時刻、完了状況
TIMESTAMP	DFT_MON_TIMESTAMP	タイム・スタンプ

スナップショットをキャプチャーしたり、イベント・モニターを使用したりする前に、データベース・マネージャーに収集させる必要のあるデータを指定する必要があります。次の特殊タイプ・データのいずれかをスナップショットで収集する場合には、該当するモニター・スイッチを設定する必要があります。

- バッファー・プール・アクティビティー情報
- ロック、ロック待機、および時間関連のロック情報
- ソート情報
- SQL ステートメント情報
- 表アクティビティー情報
- 時間およびタイム・スタンプ情報
- 作業単位情報

上記の情報タイプに対応するスイッチは、デフォルトではすべて OFF になっています。ただし、時間およびタイム・スタンプ情報に対応するスイッチだけは、デフォルトで ON になっています。

イベント・モニターは、時間およびタイム・スタンプ情報スイッチによってのみ影響を受けます。その他のすべてのスイッチ設定は、イベント・モニターによって収集されるデータには影響しません。

CLP からのシステム・モニター・スイッチの設定

システム・モニター・スイッチは、システム・モニターによるデータの収集を制御します。 特定のモニター・スイッチを ON に設定することにより、特定のタイプのモニター・データを収集できます。

モニター・スイッチの更新を実行するアプリケーションには、インスタンス・アタッチメントがなければなりません。次のコマンドを使用するには、SYSADM、SYSCTRL、SYSMAINT、または SYSMON のいずれかの権限が必要です。

- UPDATE MONITOR SWITCHES
- GET MONITOR SWITCHES
- GET DATABASE MANAGER MONITOR SWITCHES

UPDATE DBM CFG コマンドを使用するには、SYSADM 権限を持っている必要があります。

- ローカル・モニター・スイッチを活動化するには、UPDATE MONITOR SWITCHES コマンドを使用する。アプリケーション (CLP) がデタッチするか、または別の UPDATE MONITOR SWITCHES コマンドによってスイッチが非活動化されるまで、スイッチはアクティブのままです。次の例では、ローカル・モニター・スイッチをすべて ON に更新します。

  ```
  db2 update monitor switches using BUFFERPOOL on LOCK on
      SORT on STATEMENT on TIMESTAMP on TABLE on UOW on
  ```

- ローカル・モニター・スイッチを非活動化するには、UPDATE MONITOR SWITCHES コマンドを使用する。次の例では、ローカル・モニター・スイッチをすべて OFF に更新します。

  ```
  db2 update monitor switches using BUFFERPOOL off, LOCK off,
      SORT off, STATEMENT off, TIMESTAMP off, TABLE off, UOW off
  ```

  以下は、上記の UPDATE MONITOR SWITCH コマンドを発行した後に表示される出力例です。

  ```
           Monitor Recording Switches

  Switch list for db partition number 1
  Buffer Pool Activity Information (BUFFERPOOL) = OFF
  Lock Information                         (LOCK) = OFF
  Sorting Information                      (SORT) = OFF
  SQL Statement Information           (STATEMENT) = OFF
  Table Activity Information              (TABLE) = OFF
  Unit of Work Information                  (UOW) = OFF
  Get timestamp information           (TIMESTAMP) = OFF
  ```

- モニター・スイッチをデータベース・マネージャー・レベルで操作することもできる。これには、UPDATE DBM CFG コマンドを使用して、データベース・マネージャー構成ファイル内の dft_monswitches パラメーターを変更することが必要となります。以下の例では、基本情報に加えて、ロック・スイッチによって制御される情報だけが収集されます。

  ```
  db2 update dbm cfg using DFT_MON_LOCK on
  ```

  モニター・アプリケーションが開始されると、必ずデータベース・マネージャーからそのモニター・スイッチ設定を継承します。データベース・マネージャーのモニター・スイッチ設定を変更しても、実行中のモニター・アプリケーションには影響を与えません。モニター・アプリケーションは、自分自身をインスタンスに再度アタッチして、モニター・スイッチ設定の変更を取り出す必要があります。

- パーティション・データベース・システムの場合、特定のパーティションについてモニター・スイッチを固有に設定するか、またはすべてのパーティションについてグローバルに設定することができる。

  1. 特定のパーティション (例えば、パーティション番号 3) に対してモニター・スイッチ (例えば、BUFFERPOOL) を設定するには、次のコマンドを発行します。

     ```
     db2 update monitor switches using BUFFERPOOL on
         at dbpartitionnum 3
     ```

  2. すべてのパーティションについてモニター・スイッチ (例えば、SORT) を設定するには、次のコマンドを発行します。

```
db2 update monitor switches using SORT on global
```

- ローカル・モニター・スイッチの状況をチェックするには、GET MONITOR SWITCHES コマンドを使用する。

  ```
  db2 get monitor switches
  ```

- パーティション・データベース・システムの場合、特定のパーティションについてモニター・スイッチ設定を固有に表示するか、またはすべてのパーティションについてグローバルな設定を表示することができる。

  1. 特定のパーティション (例えば、パーティション番号 2) に対してモニター・スイッチ設定を表示するには、次のコマンドを発行します。

     ```
     db2 get monitor switches at dbpartitionnum 2
     ```

  2. すべてのパーティションについてのモニター・スイッチ設定を表示するには、次のコマンドを発行します。

     ```
     db2 get monitor switches global
     ```

- データベース・マネージャー・レベル (またはインスタンス・レベル) でモニター・スイッチの状況をチェックするには、 GET DATABASE MANAGER MONITOR SWITCHES コマンドを使用する。このコマンドは、モニター対象となっているインスタンスのスイッチ設定全体を表示します。

  ```
  db2 get database manager monitor switches
  ```

  以下は、上記のコマンドを発行した後に表示される出力例です。

  ```
           DBM System Monitor Information Collected

  Switch list for db partition number 1
  Buffer Pool Activity Information (BUFFERPOOL) = OFF
  Lock Information                         (LOCK) = ON    10-25-2001 16:04:39
  Sorting Information                      (SORT) = OFF
  SQL Statement Information             (STATEMENT) = OFF
  Table Activity Information               (TABLE) = OFF
  Unit of Work Information                   (UOW) = OFF
  Get timestamp information             (TIMESTAMP) = OFF
  ```

これで、目的のモニター・スイッチを設定し、スイッチ設定を確認したので、モニター・データのキャプチャーおよび収集の準備ができました。

クライアント・アプリケーションからのシステム・モニター・スイッチの設定

システム・モニター・スイッチは、システム・モニターによるデータの収集を制御します。 特定のモニター・スイッチを ON に設定することにより、特定のタイプのモニター・データを収集できます。

モニター・スイッチの更新を実行するアプリケーションには、インスタンス・アタッチメントがなければなりません。 db2MonitorSwitches API を使用するには、SYSADM、SYSCTRL、SYSMAINT、または SYSMON 権限が必要です。

1. 次の DB2 ライブラリー sqlutil.h および db2ApiDf.h を組み込みます。これらは sqllib の下の include サブディレクトリーにあります。

   ```
   #include <sqlutil.h>
   #include <db2ApiDf.h>
   #include <string.h>
   #include <sqlmon.h>
   ```

2. スイッチ・リストのバッファー単位サイズを 1 KB に設定します。

```
#define SWITCHES_BUFFER_UNIT_SZ 1024
```

3. sqlca、db2MonitorSwitches、および sqlm_recording_group 構造体を初期化します。また、スイッチ・リスト・バッファーを含むようにポインターを初期化し、バッファーのサイズを設定します。

```
struct sqlca sqlca;
memset (&sqlca, '¥0', sizeof(struct sqlca));
db2MonitorSwitchesData switchesData;
memset (&switchesData, '¥0', sizeof(switchesData));
struct sqlm_recording_group switchesList[SQLM_NUM_GROUPS];
memset(switchesList, '¥0', sizeof(switchesList));
sqluint32 outputFormat;
static sqluint32 switchesBufferSize = SWITCHES_BUFFER_UNIT_SZ;
char *switchesBuffer;
```

4. スイッチ・リスト出力を保持するバッファーを初期化します。

```
switchesBuffer = (char *)malloc(switchesBufferSize);
memset(switchesBuffer, '¥0', switchesBufferSize));
```

5. ローカル・モニター・スイッチの状態を変更するには、 sqlm_recording_group 構造体内のエレメント (前のステップでは switchesList という名前) を変更します。モニター・スイッチをオンにするには、パラメーター input_state を SQLM_ON に設定する必要があります。モニター・スイッチを OFF にするには、パラメーター input_state を SQLM_OFF に設定する必要があります。

```
switchesList[SQLM_UOW_SW].input_state = SQLM_ON;
switchesList[SQLM_STATEMENT_SW].input_state = SQLM_ON;
switchesList[SQLM_TABLE_SW].input_state = SQLM_ON;
switchesList[SQLM_BUFFER_POOL_SW].input_state = SQLM_OFF;
switchesList[SQLM_LOCK_SW].input_state = SQLM_OFF;
switchesList[SQLM_SORT_SW].input_state = SQLM_OFF;
switchesList[SQLM_TIMESTAMP_SW].input_state = SQLM_OFF;
switchesData.piGroupStates = switchesList;
switchesData.poBuffer = switchesBuffer;
switchesData.iVersion = SQLM_DBMON_VERSION9_5;
switchesData.iBufferSize = switchesBufferSize;
switchesData.iReturnData = 0;
switchesData.iNodeNumber = SQLM_CURRENT_NODE;
switchesData.poOutputFormat = &outputFormat;
```

注: iVersion が SQLM_DBMON_VERSION8 より小さい場合には、SQLM_TIMESTAMP_SW を使用できません。

6. スイッチ設定の変更を実行するには、db2MonitorSwitches() 関数を呼び出します。 db2MonitorSwitches API に対するパラメーターとして、db2MonitorSwitchesData (この例では switchesData) 構造体を渡します。switchesData には、パラメーターとして sqlm_recording_group 構造体が含まれています。

```
db2MonitorSwitches(db2Version810, &switchesData, &sqlca);
```

7. スイッチ・リスト・バッファーからのスイッチ・リスト・データ・ストリームを処理します。

8. スイッチ・リスト・バッファーをクリアします。

```
free(switchesBuffer);
free(pRequestedDataGroups);
```

これで、目的のモニター・スイッチを設定し、スイッチ設定を確認したので、モニター・データのキャプチャーおよび収集の準備ができました。

システム・モニター・スイッチ自己記述型データ・ストリーム

現行のシステム・モニター・スイッチ設定値を db2MonitorSwitches API を使って更新または表示すると、この API はスイッチ設定値を自己記述型データ・ストリームとして戻します。 図 3 では、パーティション・データベース環境の場合に戻されるスイッチ・リスト情報の構造を示します。

注:

1. これらの例や表では、ID として記述名を使用しています。これらの名前は、実際のデータ・ストリームでは **SQLM_ELM_** という接頭部が付きます。例えば db_event は、イベント・モニター出力では SQLM_ELM_DB_EVENT と表示されます。タイプは、実際のデータ・ストリームでは **SQLM_TYPE_** という接頭部が付きます。例えば、ヘッダーはデータ・ストリームで SQLM_TYPE_HEADER と表示されます。
2. グローバル・スイッチ要求では、それぞれのスイッチ要求ごとに、情報が戻されるパーティションの順番が異なる可能性があります。この場合、パーティション ID がデータ・ストリームに組み込まれます。

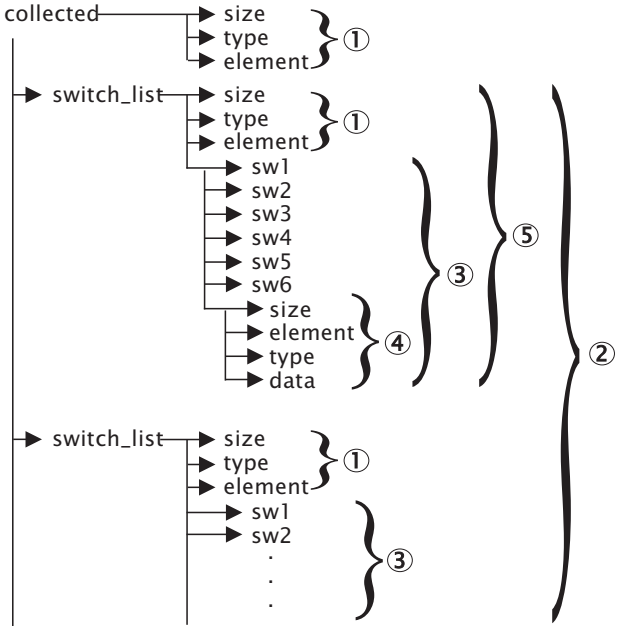


図 3. スイッチ・リスト・モニターのデータ・ストリーム

1. 各論理データ・グループは、サイズと名前を示すヘッダーで始まります。このサイズには、ヘッダー自体に取られるデータ・ボリュームは組み込まれません。
2. 収集したヘッダー内のサイズは、すべてのパーティションのすべてのモニター・スイッチ・リストの合計サイズを戻します。
3. スイッチ・リストのヘッダー内にあるサイズ・エレメントは、そのパーティションのスイッチ・データのサイズを表します。
4. スイッチ情報は、自己記述型です。
5. 非パーティション・データベースでは、独立型パーティションのスイッチ設定値が戻されます。つまり、スイッチ・リストが 1 つだけ戻されます。

データベース・システム・モニターのデータ編成

システム・モニターは情報を収集し保管します。情報はスナップショット・モニターおよび一部のイベント・モニターへのインターフェースを使用してアクセスできます。 データベース・システム・モニターは、収集した情報を、モニター・エレメント と呼ばれるエンティティーに保管します (モニター・エレメントは、以前にはデータ・エレメントと呼ばれていました)。各モニター・エレメントは、データベース・システムの状態の特定のある局面に関する情報を保管します。さらに、モニター・エレメントは固有の名前によって識別され、特定のタイプの情報を保管します。

モニター・エレメントがデータを保管するのに、システム・モニターが使用する使用可能なエレメント・タイプには、次のものがあります。

カウンター
あるアクティビティーが起こった回数を数えます。モニター中は、カウンター値は増加します。大部分のカウンター・エレメントはリセットできます。

ゲージ ある項目の現行値を示します。ゲージ値は、データベース・アクティビティーによって上下します (例えば、保持されているロックの数)。ゲージ・エレメントはリセットできません。

水準点 水準点は、モニターを開始してからエレメントが到達した最高 (最大) または最低 (最小) 値を示します。水準点エレメントはリセットできません。

情報 モニター・アクティビティーの参照タイプの詳細を提供します。これには、パーティション名、別名、およびパス詳細などが含まれます。情報エレメントはリセットできません。

タイム・スタンプ
1970 年の 1 月 1 日以降の経過した秒およびマイクロ秒数を提供することによって、アクティビティーが起こった日時を示します。スナップショット・モニターおよびイベント・モニターの場合、タイム・スタンプ・エレメントの収集は「タイム・スタンプ」モニター・スイッチによって制御されます。このスイッチはデフォルトでは ON ですが、データベース・インスタンスでの CPU 使用率が 100% に近づいた場合には、パフォーマンス上の理由でそれを OFF にする必要があります。タイム・スタンプ・エレメントはリセットできません。

タイム・スタンプ・エレメントの値 0 は、「利用不可」を意味します。このデータをインポートしようとすると、この値によって範囲外エラー (SQL0181) が生成されます。このエラーを回避するには、データをエクスポートする前に、値を任意の有効なタイム・スタンプ値に更新します。

時間 アクティビティーで使用された秒数およびマイクロ秒数を戻します。スナップショット・モニターおよびイベント・モニターの場合、大半時間エレメントの収集は「タイム」モニター・スイッチによって制御されます。このスイッチはデフォルトでは ON ですが、データベース・インスタンスでの CPU 使用率が 100% に近づいた場合には、パフォーマンス上の理由でそれを OFF にする必要があります。時間エレメントのいくつかはリセットできます。

モニター・エレメントは、1 つ以上の論理データ・グループのデータを収集します。論理データ・グループは、データベース・アクティビティーの特定の範囲のデータベース・システム・モニター情報を収集するモニター・エレメントの集合です。モニター・エレメントは、それが提供する情報のレベルに基づいて論理データ・グループにソートされます。例えば、スナップショット・モニター中に、ソート合計時間モニター・エレメントがデータベース (dbase)、アプリケーション (appl)、およびステートメント (stmt) 情報を戻します。したがって、このモニター・エレメントは括弧内に示した論理データ・グループに属します。

多くのモニター・エレメントは、スナップショット・モニターとイベント・モニターの両方によって使用されますが、それらはそれぞれ別個のセットの論理データ・グループを使用します。これは、スナップショットをキャプチャーできるデータベース・アクティビティーの範囲が、イベント・データを収集できるものの範囲とは異なるからです。実際、スナップショット・モニターからアクセス可能なモニター・エレメントの全体の内容は、イベント・モニターからアクセス可能なモニター・エレメントの内容とは異なっています。

カウンターの状況および可視性

システム・モニターによって収集されるモニター・エレメントには、いくつかの累積カウンターが含まれています。カウンターは、データベースまたはデータベース・マネージャーの操作時 (例えば、アプリケーションがトランザクションをコミットするたび) に累積されます。

カウンターが初期設定されるのは、その当該オブジェクトがアクティブになった時点です。例えば、データベース用に読み取られるバッファー・プール・ページ数 (基本モニター・エレメント) は、データベースが活動化されるとゼロに設定されます。

システム・モニターが収集できるカウンターには、モニター・スイッチによって制御されているものがあります。 あるモニター・スイッチが OFF の場合には、その制御下にあるモニター・エレメントはデータを収集しません。モニター・スイッチが ON になると、関連したすべてのカウンターがゼロにリセットされます。

イベント・モニターによって戻されるカウンターは、イベント・モニターを活動化するとゼロにリセットされます。

イベント・モニターのカウントは、次の 1 つの開始点以後のカウントを表します。

- データベース、表スペース、および表に対する、イベント・モニターの始動。
- 既存の接続に対する、イベント・モニターの始動。
- モニターが開始された後で確立された、アプリケーションの接続。
- モニターが開始された後の、次のトランザクション (作業単位) またはステートメントの開始。
- モニターが開始された後のデッドロックの発生。

イベント・モニターと、モニター・アプリケーション (スナップショット・モニター API を使用するアプリケーション) には、システム・モニター・データに関する独自の論理ビューがあります。したがって、カウンターのリセットや初期設定が行われる際には、そのリセットや初期設定を行うイベント・モニターかアプリケーシ

ョンだけが関係することになります。イベント・モニター・カウンターをリセットするには、イベント・モニターをいったん OFF にしてから ON にしなければなりません。スナップショットをとるアプリケーションの場合、 RESET MONITOR コマンドを使用するといつでもカウンターのビューをリセットできます。

ステートメントが開始された後でステートメントのイベント・モニターを開始すると、モニターは次の SQL ステートメントの開始時に情報の収集を開始します。その結果、モニターの開始時にデータベース・マネージャーが実行していたステートメントについての情報は、イベント・モニターは戻しません。これはトランザクション情報についても同様です。

システム・モニター出力 : 自己記述型データ・ストリーム

システム・モニター・データを画面に表示したり、SQL 表に保管したりすることのほかに、それを処理するクライアント・アプリケーションを開発することができます。システム・モニターは、スナップショット・モニターとイベント・モニターの両方について、自己記述型データ・ストリームを介してモニター・データを戻します。スナップショット・モニター・アプリケーションでは、スナップショット API を呼び出してスナップショットをキャプチャーした後、そのデータ・ストリームを直接処理することができます。

イベント・モニター・データの処理は、これとは異なり、データベース・イベントが発生するたびにイベント・データがアプリケーションに送信されます。パイプ・イベント・モニターの場合は、アプリケーションはイベント・データの到着を待機し、到着時にそれを処理します。ファイル・イベント・モニターの場合は、アプリケーションはイベント・ファイルを解析するため、イベント・レコードをバッチで処理します。

この自己記述型データ・ストリームによって、戻りデータを、一度に 1 つのエレメントずつ解析することができます。このことは、特定のアプリケーションや特定のデータベース状態に関する情報の検索など、モニターに関連した数多くの可能性を実現させるものとなります。

戻されるモニター・データは以下の形式になります。

size モニター・エレメントまたは論理データ・グループに保管されているデータのサイズ (バイト)。論理データ・グループの場合、これは論理グループ内の全データのサイズです。例として、データベース論理グループ (*db*) には、個々のモニター・エレメント (*total_log_used* など) やロールフォワード情報 (*rollforward*) のようなほかの論理データ・グループが含まれます。これには、'size'、'type'、および 'element' 情報に取られるサイズは組み込まれません。

type データに保管されたエレメントのタイプ (例えば、可変長ストリングまたは符号付き 32 ビット数値)。 *header* のエレメント・タイプは、エレメントの論理データ・グループを示します。

element id
モニターによってキャプチャーされたモニター・エレメントの ID。論理データ・グループの場合、これはグループの ID です (例えば、 *collected*、*dbase*、または *event_db*)。

データ あるモニター・エレメントに対して、モニターによって収集された値。論理データ・グループの場合、データはそれに属するモニター・エレメントから成っています。

モニター・エレメントのすべてのタイム・スタンプは、 2 つの符号なし 4 バイト・モニター・エレメント (秒およびマイクロ秒) で戻されます。これらは GMT (グリニッジ標準時) で 1970 年 1 月 1 日以降の秒数で表されます。

モニター・エレメント内のストリングのサイズ・エレメントは、ストリング・エレメントの実際のデータ・サイズを表します。このサイズには、NULL 終止符は入っていません。ストリングが NULL で終了することはないからです。

モニター・データのメモリー要件

モニター・データに必要なメモリーは、モニター・ヒープから割り当てられます。モニター・ヒープのサイズを制御するには **mon_heap_sz** データベース構成パラメーターを使用します。 このパラメーターのデフォルト値は AUTOMATIC であるため、instance_memory 限度に達するまで、モニター・ヒープが必要に応じて増加します。

手動で **mon_heap_sz** パラメーターを構成する場合には、次の要因を考慮に入れてください。

- モニター・アプリケーションの数
- イベント・モニターの数と性質
- 設定されたモニター・スイッチ
- データベース・アクティビティーのレベル

モニター・コマンドが SQLCODE -973 で失敗するときは、**mon_heap_sz** パラメーターの値を大きくすることを検討してください。

次の公式を使うと、モニター・ヒープに必要なページの概数を求めることができます。

```
(アプリケーションが使用するメモリー              +
 イベント・モニターが使用するメモリー            +
 モニター・アプリケーションが使用するメモリー    +
 ゲートウェイ・アプリケーションが使用するメモリー)      / 4096
```

アプリケーションごとに使用するメモリー

- STATEMENT スイッチが OFF の場合はゼロ。
- STATEMENT スイッチが ON の場合:
  - 同時に実行されるステートメントごとに 400 バイトを追加する。 (つまりアプリケーションが持つ可能性があるオープン・カーソルの数) これはアプリケーションが実行するステートメントの累計ではありません。
  - パーティション・データベースの場合は、ステートメントごとに次を追加する。
    - 200 バイト * (サブセクションの平均数)
- アプリケーションが sqleseti() 情報を発行する場合は、ユーザー ID、アプリケーション名、ワークステーション名、およびアカウンティング・ストリングのサイズを追加する。

イベント・モニターごとに使用するメモリー

イベント・モニターのタイプが ACTIVITIES の場合:

- 3500 バイト
- イベント・モニターのタイプが TABLES の場合、36K * (CPU のコア数 + 1) を追加する。
- イベント・モニターのタイプが FILE または PIPE の場合、2K * (CPU のコア数 + 1) を追加する。

容量が大きくなることが予想される場合には、イベント・レコード用に 250 MB を追加してください。 あるいは、予想される作業量に合わせて分数を追加してください。

イベント・モニターのタイプが LOCKING または UOW の場合:

- 3500 バイト
- 3K * (CPU のコア数 + 1)

容量が大きくなることが予想される場合には、イベント・レコード用に 250 MB を追加してください。 あるいは、予想される作業量に合わせて分数を追加してください。

イベント・モニターのタイプが
DATABASE、TABLES、TABLESPACES、BUFFERPOOLS、CONNECTIONS、または DEADLOCK の場合:

- 4100 バイト
- 2 * バッファー・サイズ
- イベント・モニターがファイルに書き込まれる場合は、550 バイトを追加する。
- イベント・モニターのタイプが「データベース」の場合:
  - 6000 バイトを追加
  - ステートメント・キャッシュ内のステートメントごとに 100 バイトを追加
- イベント・モニターのタイプが「表」の場合:
  - 1500 バイトを追加
  - アクセスされる表ごとに 70 バイトを追加
- イベント・モニターのタイプが「表スペース」の場合:
  - 450 バイトを追加
  - 表スペースごとに 350 バイトを追加
- イベント・モニターのタイプが「バッファー・プール」の場合:
  - 450 バイトを追加
  - バッファー・プールごとに 340 バイトを追加
- イベント・モニターのタイプが「接続」の場合:
  - 1500 バイトを追加
  - 接続されるアプリケーションごとに:
    - 750 バイトを追加
  - 137 ページの『アプリケーションごとに使用するメモリー』からの値を追加することを忘れない。

- イベント・モニターのタイプが DEADLOCK の場合:
  - および WITH DETAILS HISTORY が実行中の場合:
    - X*475 バイトに、実行中であると予想される同時アプリケーションの最大数を掛けたものを追加。ここで、X はアプリケーションの作業単位内で予想されるステートメントの最大数。
  - および WITH DETAILS HISTORY VALUES が実行中の場合:
    - X*Y バイトに、実行中であると予想される同時アプリケーションの最大数を掛けたものを追加。ここで、Y は SQL ステートメントに結び付けられているパラメーター値の予想される最大サイズ。

モニター・アプリケーションごとに使用するメモリー

- 250 バイト
- リセットされるデータベースごとに:
  - 350 バイト
  - リモート・データベースごとに 200 バイトを追加
  - SORT スイッチが ON の場合は 25 バイトを追加
  - LOCK スイッチが ON の場合は 25 バイトを追加
  - TABLE スイッチが ON の場合:
    - 600 バイトを追加
    - アクセスされる表ごとに 75 バイトを追加
  - BUFFERPOOL スイッチが ON の場合:
    - 300 バイトを追加
    - アクセスされる表スペースごとに 250 バイトを追加
    - アクセスされるバッファー・プールごとに 250 バイトを追加
  - STATEMENT スイッチが ON の場合:
    - 2100 バイトを追加
    - ステートメントごとに 100 バイトを追加
  - データベースに接続されるアプリケーションごとに:
    - 600 バイトを追加
    - アプリケーションの接続先のリモート・データベースごとに 200 バイトを追加
    - SORT スイッチが ON の場合は 25 バイトを追加
    - LOCK スイッチが ON の場合は 25 バイトを追加
    - BUFFERPOOL スイッチが ON の場合は 250 バイトを追加
- リセットされる DCS データベースごとに:
  - そのデータベース用に 200 バイトを追加
  - そのデータベースに接続されるアプリケーションごとに 200 バイトを追加
  - STATEMENT スイッチが ON で、伝送レベルのデータがリセットされる場合:
    - データベースごとに、伝送レベルごとに 200 バイトを追加
    - アプリケーションごとに、伝送レベルごとに 200 バイトを追加

ゲートウェイ・アプリケーションが使用するメモリー

- ホスト・データベースごとに 250 バイト (すべてのスイッチが OFF の場合でも)
- アプリケーションごとに 400 バイト (すべてのスイッチが OFF の場合でも)
- STATEMENT スイッチが ON の場合:
  - アプリケーションごとに、同時に実行されるステートメント (つまりアプリケーションが持つ可能性があるオープン・カーソルの数) ごとに 200 バイトを追加する。これはアプリケーションが実行するステートメントの累計ではありません。
  - 伝送レベルのデータは次のように計算する。
    - データベースごとに、伝送レベルごとに 200 バイトを追加
    - アプリケーションごとに、伝送レベルごとに 200 バイトを追加
- UOW スイッチが ON の場合:
  - アプリケーションごとに 50 バイトを追加
- TMDB (SYNCPOINT TWOPHASE アクティビティー用) を使用するアプリケーションごとに:
  - 20 バイトにさらに XID 自体のサイズを追加
- sqleseti を発行してクライアント名、アプリケーション名、ワークステーション、またはアカウンティングを設定するアプリケーションの場合:
  - 800 バイトにさらにアカウンティング・ストリング自体のサイズを追加

バッファー・プール・アクティビティーのモニター

データベース・サーバーは、バッファー・プールのすべてのデータについて読み取りと更新を行います。アプリケーションの要求に応じて、データはディスクからバッファー・プールにコピーされます。

ページは、次のようにバッファー・プール内に置かれます。

- エージェントによる。同期入出力です。
- 入出力サーバー (プリフェッチャー) による。非同期入出力です。

ページは、次のようにバッファー・プールからディスクに書き込まれます。

- エージェントにより、同期で。
- ページ・クリーナーにより、非同期で。

サーバーが 1 つのデータ・ページを読み取る必要があり、そのページがすでにバッファー・プール内にある場合は、ページをディスクから読み取るときよりも高速にそのページにアクセスできます。バッファー・プール内でできるだけたくさんのページを**ヒット**することが必要になります。ディスク I/O の回避はデータベースのパフォーマンスにおいて重要な要因であるため、バッファー・プールを適切に構成することが、パフォーマンスの調整について最も重要な考慮事項の 1 つになります。

バッファー・プールのヒット率は、データベース・マネージャーがページ要求を処理するときに、そのページがすでにバッファー・プール内に存在したためにディスクからページをロードする必要がなかった場合のパーセンテージを示します。バッファー・プール・ヒット率が高いほど、ディスク入出力の頻度は低くなります。

全体のバッファー・プール・ヒット率は、次のように計算できます。

```
    1 - ((pool_data_p_reads + pool_xda_p_reads +
pool_index_p_reads + pool_temp_data_p_reads
+ pool_temp_xda_p_reads + pool_temp_index_p_reads )
     / (pool_data_l_reads + pool_xda_l_reads + pool_index_l_reads +
     pool_temp_data_l_reads + pool_temp_xda_l_reads
     + pool_temp_index_l_reads )) * 100%
```

この計算には、バッファー・プールによってキャッシュされているすべてのページ (索引とデータ) が含まれます。

BP_HITRATIO 管理ビューを、バッファー・プールのヒット率をモニターする便利な方法として使用することもできます。

大規模なデータベースでは、バッファー・プールを大きくしても、バッファー・プール・ヒット率の効果が得られないことがあります。データ・ページ数が大きすぎるために、そのサイズを大きくしてもヒットの統計的な確率が高くならないことがあります。代わりに、索引のバッファー・プール・ヒット率を調整すると、よい結果を得られることがあります。これには 2 つの方法があります。

1. データと索引を 2 つの異なるバッファー・プールに分割し、それぞれを個別に調整します。
2. 1 つのバッファー・プールを使用し、索引のヒット率がこれ以上高くならないというところまでそのバッファー・プールのサイズを大きくします。索引のバッファー・プール・ヒット率は、次のように計算できます。

   ```
   (1 - ((pool_index_p_reads) / (pool_index_l_reads))) * 100%
   ```

最初の方法のほうが効果が上がることが多いのですが、索引とデータを別の表スペースに置く必要があるので、既存データベースには利用できないことがあります。さらに、1 つではなく、2 つのバッファー・プールを調整する必要があるので、特にメモリーに制約があるときなどは困難な作業となります。

さらに、プリフェッチャーのヒット率に対する影響を考慮する必要があります。プリフェッチャーは、アプリケーションの必要性を予想して、データ・ページをバッファー・プール内に読み取ります (非同期)。多くの場合、これらのページは必要になる直前に読み込まれます (理想的な場合)。ただし、プリフェッチャーは、使用されないページをバッファー・プール内に読み込むことで不要な入出力を行うこともあります。例えば、あるアプリケーションが表の読み取りを開始するとします。このことが検出されると、プリフェッチが開始しますが、アプリケーションはアプリケーション・バッファーを充てんして読み取りを停止します。この間に、その他の多数のページがプリフェッチされます。結局使用されることのないページについて入出力が行われ、バッファー・プールの一部がこうしたページで占められることになってしまいます。

ページ・クリーナーは、バッファー・プールをモニターし、ディスクにページを非同期で書き込みます。これには次の目的があります。

- エージェントがバッファー・プール内でフリー・ページを必ず見つけられるようにする。エージェントがバッファー・プール内でフリー・ページを見つけられない場合は、エージェント自身がページをクリーニングしなければならないため、関連アプリケーションに対してよい応答を返せないことになります。

- システムがクラッシュした場合に、データベースのリカバリーを迅速に行う。ディスクに書き込まれたページ数が多いほど、データベースのリカバリーで処理が必要となるログ・ファイル・レコードの数は少なくなります。

ダーティー・ページはディスクに書き出されますが、バッファー・プールに新しいページを読み取るためのスペースが必要な場合を除いて、このページはバッファー・プールからすぐには除去されません。

注: バッファー・プールに関する情報は、通常は表スペース・レベルで収集されますが、データベース・システム・モニターの機能により、この情報はバッファー・プールおよびデータベースのレベルにまでまとめることができます。実行する分析の種類にもよりますが、いずれかのレベルまたはすべてのレベルでこのデータを調査する必要があります。

データベース・システム・モニター・インターフェース

モニター・タスク	API
スナップショットのキャプチャー	db2GetSnapshot
自己記述型データ・ストリームの変換	db2ConvMonStream
データベース・システム・モニター・スイッチの表示	db2MonitorSwitches
スナップショットのサイズ見積もり	db2GetSnapshotSize
モニター・スイッチの取得/更新	db2MonitorSwitches
モニター・カウンターのリセット	db2ResetMonitor
データベース・システム・モニター・スイッチの更新	db2MonitorSwitches

モニター・タスク	CLP コマンド
イベント・モニター出力の GUI ツールによる分析	db2eva
スナップショットのキャプチャー	GET SNAPSHOT
データベース・マネージャー・モニター・スイッチの表示	GET DATABASE MANAGER MONITOR SWITCHES
モニター・アプリケーションのモニター・スイッチの表示	GET MONITOR SWITCHES
イベント・モニター・トレースのフォーマット	db2evmon
表書き込み CREATE EVENT MONITOR ステートメントに関する SQL 例の生成	db2evtbl
アクティブなデータベースのリスト	LIST ACTIVE DATABASES
データベースに接続されたアプリケーションのリスト	LIST APPLICATIONS
DCS アプリケーションのリスト	LIST DCS APPLICATIONS
モニター・カウンターのリセット	RESET MONITOR
データベース・システム・モニター・スイッチの更新	UPDATE MONITOR SWITCHES

モニター・タスク	SQL ステートメント
イベント・モニターの活動化	SET EVENT MONITOR STATE
イベント・モニターの作成	CREATE EVENT MONITOR
イベント・モニターの非活動化	SET EVENT MONITOR STATE
イベント・モニターの削除	DROP
イベント・モニター値の書き込み	FLUSH EVENT MONITOR

モニター・タスク	SQL 関数
イベント・モニターの状態の判別	EVENT_MON_STATE スカラー関数
データベース・マネージャー・レベルのスナップショットの取得	SNAPDBM 管理ビューおよび SNAP_GET_DBM_V95 表関数
データベース・マネージャー・レベルでの、現行のモニター・スイッチ設定値の取得	SNAPSWITCHES 管理ビューおよび SNAP_GET_SWITCHES 表関数
高速コミュニケーション・マネージャーのスナップショットの取得	SNAPFCM 管理ビューおよび SNAP_GET_FCM 表関数
指定されたパーティションについての、高速コミュニケーション・マネージャーのスナップショットの取得	SNAPFCM_PART 管理ビューおよび SNAP_GET_FCM_PART 表関数
データベース・レベルのスナップショットの取得	SNAPDB 管理ビューおよび SNAP_GET_DB_V95 表関数
アプリケーション・レベルのスナップショットの取得	SNAPAPPL 管理ビューおよび SNAP_GET_APPL_V95 表関数
アプリケーション・レベルのスナップショットの取得	SNAPAPPL_INFO 管理ビューおよび SNAP_GET_APPL_INFO_V95 表関数
ロック待機情報に関する、アプリケーション・レベルのスナップショットの取得	SNAPLOCKWAIT 管理ビューおよび SNAP_GET_LOCKWAIT 表関数
ステートメント情報に関する、アプリケーション・レベルのスナップショットの取得	SNAPSTMT 管理ビューおよび SNAP_GET_STMT 表関数
エージェント情報に関する、アプリケーション・レベルのスナップショットの取得	SNAPAGENT 管理ビューおよび SNAP_GET_AGENT 表関数
サブセクション情報に関する、アプリケーション・レベルのスナップショットの取得	SNAPSUBSECTION 管理ビューおよび SNAP_GET_SUBSECTION 表関数
バッファー・プール・レベルのスナップショットの取得	SNAPBP 管理ビューおよび SNAP_GET_BP_V95 表関数
表スペース・レベルのスナップショットの取得	SNAPTBSP 管理ビューおよび SNAP_GET_TBSP_V91 表関数
構成情報に関する、表スペース・レベルのスナップショットの取得	SNAPTBSP_PART 管理ビューおよび SNAP_GET_TBSP_PART_V91 表関数
コンテナー情報に関する、表スペース・レベルのスナップショットの取得	SNAPCONTAINER 管理ビューおよび SNAP_GET_CONTAINER_V91 表関数
静止プログラム情報に関する、表スペース・レベルのスナップショットの取得	SNAPTBSP_QUIESCER 管理ビューおよび SNAP_GET_TBSP_QUIESCER 表関数
表スペース・マップの範囲に関する、表スペース・レベルのスナップショットの取得	SNAPTBSP_RANGE 管理ビューおよび SNAP_GET_TBSP_RANGE 表関数
表レベルのスナップショットの取得	SNAPTAB 管理ビューおよび SNAP_GET_TAB_V91 表関数
ロック・レベルのスナップショットの取得	SNAPLOCK 管理ビューおよび SNAP_GET_LOCK 表関数

モニター・タスク	SQL 関数
SQL ステートメントのキャッシュ情報のスナップショットの取得	SNAPDYN_SQL 管理ビューおよび SNAP_GET_DYN_SQL_V95 表関数

第 6 章 非推奨のモニター・ツール

ヘルス・モニター

データベースの正常性のモニター

ヘルス・モニターの概要

ヘルス・モニターとは、サーバー・サイドのツールの一種で、インスタンスとアクティブ・データベースの正常性を定期的にモニターして、例外による管理機能を追加します。ヘルス・モニターには、潜在的なシステムの正常性の問題についてデータベース管理者 (DBA) にアラートを出す機能もあります。ヘルス・モニターは、ハードウェア障害や、受け入れがたいシステム・パフォーマンスまたは機能の低下を引き起こしかねない問題を事前に検出します。ヘルス・モニターには事前の対策を講じる性質があるので、ユーザーは、システム・パフォーマンスに影響する問題に発展する前にその問題に取り組むことができます。

ヘルス・モニターは、ヘルス・インディケーターを使用してシステムの状態を検査し、アラートを発行する必要があるかどうかを判別します。アラートに応じて、事前構成済みのアクションが取られます。さらにヘルス・モニターは、管理通知ログにアラートを記録し、電子メールまたはページャーで通知を送信することもできます。この例外による管理モデルにより、アクティブ・モニターを必要とせずに、潜在的なシステムの正常性に関する問題に対するアラートを生成できるので、貴重な DBA リソースを解放できます。

ヘルス・モニターは、全体のパフォーマンスにほとんど影響を与えずに、システムの正常性に関する情報を定期的に収集します。情報を収集するのに、スナップショット・モニター・スイッチを ON にしません。

ヘルス・インディケーター:

ヘルス・モニターは、ヘルス・インディケーターを使用して、データベース・マネージャーやデータベースのパフォーマンスの特定の性質の正常性を評価します。ヘルス・インディケーターは、表スペースなど特定のクラスのデータベース・オブジェクトのある性質の正常性を測定します。正常性を判別するため、測定値に対してある基準が適用されます。ここで適用される基準は、ヘルス・インディケーターのタイプに従属します。この基準に基づいて正常稼働ではないと判別されると、アラートが生成されます。

ヘルス・モニターによって、以下の 3 つのタイプのヘルス・インディケーターが戻されます。

- **しきい値ベースの**インディケーターは、オブジェクトの動作の (連続的な範囲の値の) 統計を表すメジャーである。警告およびアラームしきい値は、正常、警告、およびアラーム範囲の境界つまりゾーンを定義します。しきい値ベースのヘルス・インディケーターには、正常、警告、およびアラームの 3 つの有効な状態があります。

- **状態ベースの**インディケーターは、データベース・オブジェクトまたはリソースの操作が正常かどうかを定義するオブジェクトの、 2 つ以上の異なる状態の限定集合を表すメジャーである。これらの状態の 1 つが通常で、その他の状態はすべて通常ではないと見なされます。状態ベースのヘルス・インディケーターには、通常およびアテンションの 2 つの有効な状態があります。
- **コレクション状態ベースの**インディケーターは、データベース中の 1 つ以上のオブジェクトのコレクション状態を表すデータベース・レベルのメジャーである。コレクション中のオブジェクトごとにデータが取り込まれ、これらのオブジェクトの間で最も重大な条件が集約状態として表されます。コレクション中の 1 つ以上のオブジェクトがアラートを必要とする状態である場合は、ヘルス・インディケーターはアテンションを表示します。コレクション状態ベースのヘルス・インディケーターには、通常およびアテンションの 2 つの有効な状態があります。

ヘルス・インディケーターは、インスタンス、データベース、表スペース、および表スペース・コンテナー・レベルです。

ヘルス・モニター情報には、ヘルス・センター、CLP、または API を介してアクセスできます。これらのツールを使用してヘルス・インディケーターの構成も行えます。

アラートは、正常の状態から正ではない状態への変化、または定義されたしきい値の境界に基づくヘルス・インディケーター値の警告またはアラームのゾーンへの変化に応答して生成されます。アラートには、アテンション、警告、およびアラームの 3 つがあります。

- 特定の状態を測定するヘルス・インディケーターの場合、通常でない状態が登録されると、アテンション・アラートが発行される。
- 値の連続範囲を計測するヘルス・インディケーターの場合は、正常、警告、およびアラームの各状態の境界やゾーンは、しきい値によって定義される。例えば、値がアラーム・ゾーンとして定義されている値のしきい値範囲に入ると、アラームのアラートが発行されて、問題に対して即時に対処が必要であることを示します。

ヘルス・モニターは、特定のヘルス・インディケーターの特定のアラート条件が最初に現れたときにのみ通知を送信し、アクションを実行します。ヘルス・インディケーターが特定のアラート条件のまま留まる場合、追加の通知は送信されず、追加のアクションも実行されません。ヘルス・インディケーターがアラート条件を変更するか、または通常の状態に戻って再びアラート条件に入る場合、新たに通知が送信され、アクションが実行されます。

以下の表は、様々なリフレッシュ・インターバルにおけるヘルス・インディケーターと、ヘルス・インディケーターの状態に対するヘルス・モニターの応答の例を示しています。この例では、デフォルトの警告しきい値として 80 %、アラームしきい値として 90 % を使用しています。

表 45. 様々なリフレッシュ・インターバルにおけるヘルス・インディケーターの状態

リフレッシュ・インターバル	ts.ts_util (表スペースの使用率) ヘルス・インディケーターの値	ts.ts_util ヘルス・インディケーターの状態	ヘルス・モニターの応答
1	80	警告	警告の通知が送信され、警告アラート条件のアクションが実行される
2	81	警告	通知は送信されず、アクションは実行されない
3	75	正常	通知は送信されず、アクションは実行されない
4	85	警告	警告の通知が送信され、警告アラート条件のアクションが実行される
5	90	アラーム	アラームの通知が送信され、アラーム条件のアクションが実行される

ヘルス・インディケーターのプロセスのサイクル:

次の図は、ヘルス・インディケーターの評価プロセスを図示しています。ステップの集合は、特定のヘルス・インディケーターのリフレッシュ・インターバルが経過するたびに実行されます。

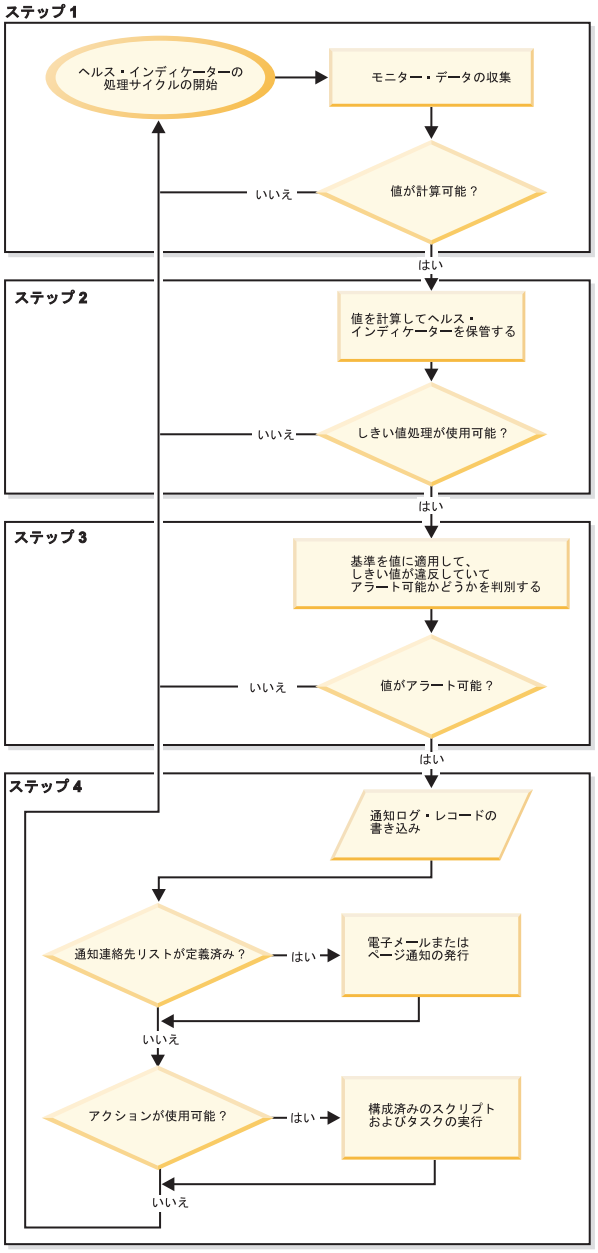


図 4. ヘルス・インディケーターのプロセスのサイクル

注:

1. NOTIFYLEVEL データベース・マネージャー構成パラメーターは、アラート通知が DB2 管理通知ログと定義済みの連絡先に送信するかどうかを制御します。アラーム通知には重大度レベル 2 以上が必要です。警告およびアテンション・アラートを送信するには、重大度レベル 3 以上が必要です。

ヘルス・アラート通知の使用可能化:

アラートの生成時に電子メールまたはページャーで通知できるようにするには、構成パラメーターを設定して連絡先情報を指定しなければなりません。

連絡先リストのあるシステム上で、 DB2 Administration Server (DAS) が実行されている必要があります。例えば、CONTACT_HOST 構成パラメーターがリモート・システムに設定されている場合は、連絡先にアラートを通知するには、リモート・システム上で DAS が実行されていなければなりません。

ヘルス・アラート通知を使用可能にするには、次のようにします。

1. SMTP_SERVER パラメーターを指定します。 DAS 構成パラメーター SMTP_SERVER は、電子メールとページャーの両方の通知メッセージを送信する際に使用するメール・サーバーの場所を指定します。 DB2 データベースのインストール先のシステムが非認証 SMTP サーバーとして使用可能になっている場合は、このステップを省略してください。
2. CONTACT_HOST パラメーターを指定します。 DAS 構成パラメーター CONTACT_HOST は、ローカル・システム上のすべてのインスタンス用の連絡先リストのリモート位置を指定します。このパラメーターを設定すると、複数のシステム間で 1 つの連絡先リストを共有できます。 DB2 データベースのインストール先のローカル・システム上に連絡先リストを維持する場合は、このステップを省略してください。
3. ヘルス・モニター通知のデフォルトの連絡先を指定します。アラートの生成時にヘルス・モニターから電子メールまたはページャー通知を行えるようにするには、デフォルトの管理連絡先を指定しなければなりません。この情報を指定しないことを選択した場合は、アラート条件に関する通知メッセージを送信できません。インストール中にデフォルトの管理連絡先情報を指定するか、またはインストールが完了するまでこの作業を遅らせることができます。 この作業を遅らせることを選択した場合や、通知リストにさらに連絡先かグループを追加する場合は、 CLP、C API、またはヘルス・センターを使用して連絡先を指定できます。

   - **CLP を使用して連絡先を指定するには、次のようにしてください。**
     ヘルス・モニター通知のデフォルトとして電子メールの連絡先を定義するには、次のコマンドを発行してください。

     ```
     DB2 ADD CONTACT contact_name TYPE EMAIL ADDRESS
        email_address DESCRIPTION 'Default Contact'

     DB2 UPDATE NOTIFICATION LIST ADD CONTACT contact_name
     ```

     完全な構文の詳細については、「コマンド・リファレンス」を参照してください。

   - **C API を使用して連絡先を指定するには、次のようにしてください。**
     次の C コードの抜粋は、正常性の通知の連絡先を定義する方法を図示しています。

     ```
     ...
     #include <db2ApiDf.h>

     SQL_API_RC rc = 0;
     struct db2AddContactData addContactData;
     struct sqlca sqlca;

     char* userid = "myuser";
     char* password = "pwd";
     char* contact = "DBA1";
     ```

```
char* email = "dba1@mail.com";
char* desc = "Default contact";

memset(&addContactData, '¥0', sizeof(addContactData));
memset (&sqlca, '¥0', sizeof(struct sqlca));

addContactData.piUserid = userid;
addContactData.piPassword = password;
addContactData.piName = contact;
addContactData.iType = DB2CONTACT_EMAIL;
addContactData.piAddress = email;
addContactData.iMaxPageLength = 0;
addContactData.piDescription = desc;

rc = db2AddContact(db2Version810, &addContactData, &sqlca);

if (rc == 0) {
   db2HealthNotificationListUpdate update;
   db2UpdateHealthNotificationListData data;
   db2ContactTypeData contact;

   contact.pName = contact;
   contact.contactType = DB2CONTACT_EMAIL;

   update.iUpdateType = DB2HEALTHNOTIFICATIONLIST_ADD;
   update.piContact = &contact;

   data.iNumUpdates = 1;
   data.piUpdates = &update;

   rc = db2UpdateHealthNotificationList (db2Version810, &data, &ca);
}
...
```

- **ヘルス・センターを使用して連絡先を指定するには、次のようにしてください。**

  a. 正常性の通知リストを定義するインスタンスを右マウス・ボタンでクリックします。

  b. 「**構成**」をクリックしてから、「**アラート通知**」をクリックします。「ヘルス・アラート通知の構成」ウィンドウが開きます。

  c. ウィンドウの左側の「**選択可能**」リストに連絡先が表示されない場合は、「**連絡先の管理**」をクリックします。「連絡先」ウィンドウが開き、システム名が事前選択されています。

  d. 「**連絡先の追加**」をクリックします。「連絡先の追加」ウィンドウが開きます。

  e. 名前と電子メール・アドレスを指定して、連絡先を定義します。ページャーの電子メール・アドレスを指定する場合は、「**アドレスはページャー用**」を選択します。

  f. 「**OK**」をクリックします。

  g. 「連絡先」ウィンドウを閉じて、「ヘルス・アラート通知の構成」ウィンドウに戻ります。この時点で、新しい連絡先が「**選択可能な連絡先**」リストに表示されています。

  h. 右矢印ボタンをクリックして、連絡先を「**ヘルス通知コンタクト・リスト**」に移動します。

i. 「**OK**」をクリックして、正常性の通知リストに連絡先を組み込みます。

推奨 通知に関する障害が生じている場合は、「ヘルス通知コンタクト・リスト」の下の「**トラブルシューティング**」を選択してください。「ヘルス・アラート通知のトラブルシューティング」ウィザードが開きます。

ヘルス・センターの概要

DB2 の稼働状態を分析および改善するには、ヘルス・センターを使用します。

DB2 が正常であると判断される状況としては、以下のようなものがあります。

- タスクを実行するために、空きメモリー、表スペース・コンテナー、またはロギング・ストレージなどのリソースが十分にある。
- リソースが効率的に使用されている。
- タスクが許容時間内に完了する、目立ったパフォーマンスの低下がない。
- リソースまたはデータベース・オブジェクトが使用不可能な状態のままにならない。

重要: ヘルス・センターは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。 詳しくは、「*DB2 バージョン 9.7 の新機能*」のトピック『コントロール・センター・ツールおよび DB2 管理サーバー (DAS) が推奨されなくなった』を参照してください。

ヘルス・センターから、他のセンターやツールを開いて、データベースの稼働状態の調査や保守に役立てることもできます。

Intel® プラットフォーム上でヘルス・センターを開くには、「**スタート**」 メニューから、「**スタート**」 → 「**プログラム**」 → 「**IBM DB2**」 → 「**モニター・ツール**」 → 「**ヘルス・センター**」 とクリックします。

Intel プラットフォーム上で、コマンド行を使用してヘルス・センターを開くには、以下のコマンドを実行します。

```
db2hc
```

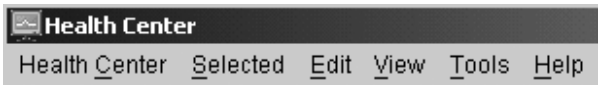
以下のリストは、ヘルス・センターを使って行える主要なタスクのいくつかを分類しています。

- 148 ページの『ヘルス・アラート通知の使用可能化』
  - 連絡先の設定および通知構成パラメーターの指定
  - ヘルス・アラート通知のトラブルシューティング
- 178 ページの『ヘルス・センターを使用したヘルス・インディケーターの構成』
  - ヘルス・インディケーター評価の使用可能化と使用不可化
  - アラートしきい値および感度の設定値の変更
  - アラートが発生した場合のタスクとスクリプトの実行
- 170 ページの『ヘルス・センターを使用したヘルス・モニター・アラートの解決』
  - 推奨アドバイザーを使用した、推奨事項の選択とインプリメント

ヘルス・センター・インターフェース

ヘルス・センター・インターフェースには、システムの稼働状態全般に関する問題を判別および解決する際に役立つ以下のエレメントがあります。

ヘルス・センター・メニュー・バー



ヘルス・センターのオブジェクトを処理したり、他の管理センターやツールを開いたり、オンライン・ヘルプにアクセスするには、メニュー・バーを使用します。

ヘルス・センター・メニュー・バーには、以下のメニューが含まれます。

ヘルス・センター・ツールバー

他のセンターおよびツールへアクセスしたり、ヘルス・センターの内容ビューをリフレッシュしたりするには、メニュー・バーの下のツールバー・アイコンを使用します。

トグル・ボタン

ナビゲーション・ビューに表示されるアラート状態を選択するには、トグル・ボタンを使用します。各ボタンは、データベース・オブジェクトをビューに表示するために必要な最低限のアラート重大度に対応します。別のボタンを選択すると、表示にのみ影響しますが、オブジェクトそのものへは影響しません。

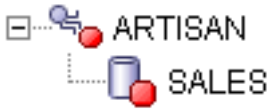
 アラーム状態のオブジェクトを表示します。

 アラームおよび警告状態のオブジェクトを表示します。

 アラート状態 (アラーム、警告、アテンション、通常、および非モニター状態) のオブジェクトを表示します。

 すべてのオブジェクトを表示します。

ナビゲーション・ビュー



インスタンスおよびデータベース・オブジェクトの表示および処理を行うには、ナビゲーション・ビューを使用します。ナビゲーション・ビューでオブ

ジェクトを選択すると、そのオブジェクトとそのすべての子に関する現行アラートがアラート・ビューに表示されます。ナビゲーション・ビューにアラートが表示される前にオブジェクトが持っていなければならないアラートのレベルを変更するには、ナビゲーション・ビューの空白部分で右クリックしてください。これにより、アラート・レベルのポップアップ・メニューが開きます。表示するアラート・レベルを選択します。トグル・ボタンをクリックして、表示するアラート・レベルを選択することもできます。

アラート・ビュー

現行アラートを表示および処理するには、アラート・ビューを使用します。アラート・ビューは、ナビゲーション・ビューで選択されたオブジェクトとその子データベース・オブジェクトに関する、現在存在しているアラートを表示します。例えば、インスタンスを選択すると、そのインスタンスと、そのインスタンスのすべてのデータベースおよび表スペースに関するアラートが表示されます。データベースを選択すると、そのデータベースとデータベースのすべての表スペースに関するアラートが表示されます。アラート・ビューのアラートに対するアクションを呼び出すには、1 つ以上のアラートを選択し、右クリックします。

アラート・ビュー・ツールバー

アラート・ビューでのアラートの表示をユーザーの要件に合うように調整するには、アラート・ビューの下にあるツールバーを使用します。

アラート条件の調査:

ヘルス・モニター

ヘルス・モニターは、データベース・マネージャー、データベース、表スペース、および表スペース・コンテナーに関する情報をキャプチャーします。ヘルス・モニターは、データベース・システム・モニター・エレメント、オペレーティング・システム、および DB2 データベースから取り出されるデータに基づいて、ヘルス・インディケーターを計算します。ヘルス・モニターがデータベースとそのオブジェクトに関するヘルス・インディケーターを評価できるのは、データベースがアクティブの場合に限ります。 ACTIVATE DATABASE コマンドを使用してデータベースを開始するか、データベースに対する永続接続を保守することにより、データベースをアクティブにしておくこともできます。

ヘルス・モニターは、ヘルス・インディケーターごとに最大 10 個の履歴レコードを保持します。この履歴は、 <instance path>¥hmonCache ディレクトリーに格納され、ヘルス・モニターの停止時に除去されます。ヘルス・モニターは、レコードの最大数に達した時点で、古い履歴レコードを自動的に整理します。

ヘルス・モニター・データにはヘルス・スナップショットを使用してアクセスできます。個々のヘルス・スナップショットは、最新のリフレッシュ・インターバルに基づいて、ヘルス・インディケーターごとに状況を報告します。スナップショットは、既存のデータベースの正常性に関する問題を検出したり、データベース環境で正常性が低くなる可能性を予測したりするのに便利です。ヘルス・スナップショッ

トは、 C または C++ アプリケーション中の API を使用して CLP からキャプチャーしたり、グラフィック管理ツールを使用することによってキャプチャーしたりすることができます。

ヘルス・モニターでは、インスタンス接続が必要です。 ATTACH TO コマンドを使用してインスタンス接続が確立されていない場合、ローカル・インスタンスへのデフォルトのインスタンス接続が作成されます。

パーティション・データベース環境では、スナップショットは、インスタンスの任意のパーティションでとることも、単一のインスタンス接続を使用してグローバルにとることもできます。グローバル・スナップショットは、それぞれのパーティションで収集されたデータを集約して単一の値セットを戻します。

使用上の注意

ヘルス・モニターは、DB2 データベースのすべてのエディションでサポートされています。

ヘルス・センターからヘルス・モニターを開始または停止するには、ヘルス・センターのナビゲーション・ビューでインスタンスを右クリックし、「ヘルス・モニターの開始」または「ヘルス・モニターの停止」を選択します。

Windows では、DB2 インスタンスのサービスは、 SYSADM 権限のあるアカウントの下で実行する必要があります。管理者特権のあるアカウントを使用するには、db2icrt コマンド上で「**-u**」オプションを使用するか、または Windows の「サービス」フォルダーを使用して、「ログオン」プロパティーを編集します。

ヘルス・モニター・プロセスは、DB2 fenced モード・プロセスとして実行されます。これらのプロセスは、Windows では DB2FMP として表示されます。他のプラットフォームでは、ヘルス・モニター・プロセスは DB2ACD として表示されます。

通知が送信され、アラート・アクションが実行されるには、ヘルス・モニターのあるシステム上で、 DB2 Administration server が実行されていなければなりません。リモートのスクリプト、タスク、または連絡先リストを使用する場合は、リモート・システム上で DB2 Administration server も開始しなければなりません。

ツール・カタログ・データベースは、タスクを作成する場合のみ必要です。ヘルス・インディケーターに関するアラート・タスク・アクションを使用しない場合は、ヘルス・モニターにはツール・カタログ・データベースは必要ありません。

DB2 UDB バージョン 8.1 より後の DB2 データベース・システム・バージョンからバージョン 8.1 にフォールバックする場合、行われたレジストリーの変更はすべて失われます。レジストリーは、バージョン 8.1 の HealthRules.reg ファイルに戻ります。このファイルには、より新しいレジストリー・ファイルの設定を使ってアップグレードおよび開始する前に存在していた設定が含まれています。

ヘルス・インディケーターのデータ: ヘルス・モニターは、個々のデータベース・パーティションに関するヘルス・インディケーターごとに、以下を含むデータの集合を記録します。

- ヘルス・インディケーター名

- 値
- 評価タイム・スタンプ
- アラート状態
- 公式 (該当する場合)
- 追加情報 (該当する場合)
- 最新のヘルス・インディケーター評価の履歴 (最大 10)。個々の履歴項目は、次のヘルス・インディケーター評価を、現行ヘルス・インディケーターの出力に至るまでキャプチャーします。
  - 値
  - 公式 (該当する場合)
  - アラート状態
  - タイム・スタンプ

ヘルス・モニターは、インスタンス、データベース、および表スペース・レベルで、最大重大度アラート状態の追跡も行います。それぞれのレベルで、このヘルス・インディケーターは、そのレベルと、そのレベルより低いすべてのレベルのヘルス・インディケーターに関する、既存の最大重大度アラートを表します。例えば、インスタンスに関する最大重大度アラート状態には、インスタンス、そのすべてのデータベース、およびそれらのデータベースごとのすべての表スペースと表スペース・コンテナーに関するヘルス・インディケーターが含まれます。

データベースのヘルス・スナップショットのキャプチャー:

SQL 表関数を使用したデータベースのヘルス・スナップショットのキャプチャー:

SQL 表関数を使用して、データベースのヘルス・スナップショットをキャプチャーすることができます。使用可能なそれぞれのヘルス・スナップショット表関数は、ヘルス・スナップショット要求タイプに対応しています。

SQL 表関数を使用して、データベースのヘルス・スナップショットをキャプチャーするには、次のようにします。

1. 使用しようとしている SQL 表関数を識別します。

   SQL 表関数には、以下の 2 つの入力パラメーターがあります。
   - VARCHAR(255): データベース名。
   - INT: パーティション番号 (0 から 999 の間の値)。モニターするパーティション番号に対応する整数を入力します。現在接続しているパーティションのスナップショットをキャプチャーする場合は、値 -1 を入力します。グローバル・スナップショットをキャプチャーするには、値 -2 を入力します。

   注: データベース・マネージャーのスナップショット SQL 表関数だけはこの規則の例外です。それには 1 つのパラメーターしかありません。この 1 つのパラメーターは、パーティション番号のパラメーターです。データベース名パラメーターに NULL を入力すると、モニターは、表関数の呼び出しに使用される接続によって定義されたデータベースを使用します。
2. SQL ステートメントを発行します。

以下の例では、現在接続されているパーティション、およびこの表関数呼び出しが行われた接続によって定義されたデータベース上についての、基本的なヘルス・スナップショットをキャプチャーします。

```
SELECT * FROM TABLE( HEALTH_DB_INFO( cast (NULL as VARCHAR(1)), -1))
       as HEALTH_DB_INFO
```

戻り表から個々のモニター・エレメントを選択することもできます。戻り表の各列は、モニター・エレメントに対応しています。したがって、各モニター・エレメントの列名は、モニター・エレメントの名前に対応しています。次のステートメントの場合は、 db_path と server_platform のモニター・エレメントだけが戻されます。

```
SELECT db_path, server_platform
       FROM TABLE( HEALTH_DB_INFO( cast (NULL as VARCHAR(1)), -1 ) )
       as HEALTH_DB_INFO
```

CLP を使用したデータベースのヘルス・スナップショットのキャプチャー:

CLP から GET HEALTH SNAPSHOT コマンドを使用して、ヘルス・スナップショットをキャプチャーすることができます。コマンド構文は、ヘルス・モニターによってモニターされたさまざまなタイプのヘルス・スナップショット情報の取り出しをサポートします。

ヘルス・スナップショットをキャプチャーするには、インスタンス接続がなければなりません。インスタンス接続がない場合、デフォルトのインスタンス接続が作成されます。リモート・インスタンスのスナップショットを取得するには、まずそのインスタンスにアタッチする必要があります。

CLP を使用して、データベースのヘルス・スナップショットをキャプチャーするには、次のようにします。

1. CLP から、GET HEALTH SNAPSHOT コマンドを必要なパラメーターを指定して発行します。

   次の例では、データベース・マネージャーの開始直後に、データベース・マネージャー・レベルのヘルス・スナップショットがキャプチャーされます。

   ```
   db2 get health snapshot for dbm
   ```

2. パーティション・データベース・システムの場合、特定のパーティションについてデータベース・スナップショットを固有にキャプチャーすることも、すべてのパーティションについてグローバルなスナップショットをキャプチャーすることもできます。特定のパーティション (例えば、パーティション番号 2) 上のデータベースのヘルス・スナップショットをキャプチャーするには、次のコマンドを発行します。

   ```
   db2 get health snapshot for db on sample at dbpartitionnum 2
   ```

   すべてのパーティション上のすべてのアプリケーションについてのデータベース・スナップショットをキャプチャーするには、次のコマンドを発行します。

   ```
   db2 get health snapshot for db on sample global
   ```

   次のコマンドは、公式、追加情報、およびヘルス・インディケーター履歴を含む追加の詳細情報のある、ヘルス・スナップショットをキャプチャーします。

   ```
   db2 get health snapshot for db on sample show detail
   ```

3. コレクション状態ベースのヘルス・インディケーターの場合、状態にかかわらず、すべてのコレクション・オブジェクトについてデータベース・スナップショットをキャプチャーすることができます。正規の GET HEALTH SNAPSHOT FOR DB コマンドは、すべてのコレクション状態ベースのヘルス・インディケーターについて、アラートが必要なすべてのコレクション・オブジェクトを戻します。

   すべてのコレクション・オブジェクトをリストしてデータベースのヘルス・スナップショットをキャプチャーするには、次のコマンドを発行します。

   ```
   db2 get health snapshot for db on sample with full collection
   ```

クライアント・アプリケーションからのデータベースのヘルス・スナップショットのキャプチャー:

C または C++ アプリケーションでスナップショット・モニター API を使用して、ヘルス・スナップショットをキャプチャーすることができます。 db2GetSnapshot API にパラメーターを指定することにより、多数の異なるヘルス・スナップショット要求タイプにアクセスすることができます。

ヘルス・スナップショットをキャプチャーするには、インスタンスにアタッチしていなければなりません。インスタンス接続がない場合、デフォルトのインスタンス接続が作成されます。リモート・インスタンスのスナップショットを取得するには、まずそのインスタンスにアタッチする必要があります。

1. コードに sqlmon.h および db2ApiDf.h DB2 ライブラリーを組み込みます。これらのライブラリーは、 sqllib¥include ディレクトリーにあります。

   ```
   #include <db2ApiDf.h>
   #include <sqlmon.h>
   ```

2. スナップショットのバッファー単位サイズを 50 KB に設定します。

   ```
   #define SNAPSHOT_BUFFER_UNIT_SZ 51200
   ```

3. sqlma、sqlca、sqlm_collected、および db2GetSnapshotData 構造体を宣言します。

   ```
   struct sqlma *pRequestedDataGroups;
   struct sqlca sqlca;
   memset (&sqlca, '¥0', sizeof(struct sqlca));
   struct sqlm_collected collectedData;
   memset (&sqlm_collected, '¥0', sizeof(struct sqlm_collected));
   db2GetSnapshotData getSnapshotParam;
   memset(&db2GetSnapshotData, '¥0', sizeof(db2GetSnapshotData));
   ```

4. スナップショット・バッファーを含み、バッファーのサイズを設定するようにポインターを初期化します。

   ```
   static sqluint32 snapshotBufferSize = SNAPSHOT_BUFFER_UNIT_SZ;
   sqluint32 outputFormat;
   char *snapshotBuffer;
   ```

5. sqlma 構造体を初期化し、キャプチャーしようとしているスナップショットがデータベース・マネージャー・レベル情報のものであることを指定します。

   ```
   pRequestedDataGroups = (struct sqlma *)malloc(SQLMASIZE(1));
   memset(&pRequestedDataGroups, '¥0', sizeof(struct pRequestedDataGroups));
   pRequestedDataGroups->obj_num = 1;
   pRequestedDataGroups->obj_var[0].obj_type = SQLMA_DB2;
   ```

6. スナップショット・リスト出力を保持するバッファーを初期化します。

```
snapshotBuffer = (char *)malloc(snapshotBufferSize);
memset (&snapshotBuffer, '¥0', sizeof(snapshotBuffer));
```

7. db2GetSnapshotData 構造体に、スナップショット要求タイプ (sqlma 構造体から)、バッファー情報、およびスナップショットをキャプチャーするために必要な他の情報を含めます。

```
getSnapshotParam.piSqlmaData = pRequestedDataGroups;
getSnapshotParam.poCollectedData = &collectedData;
getSnapshotParam.poBuffer = snapshotBuffer;
getSnapshotParam.iVersion = SQLM_DBMON_VERSION9_5;
getSnapshotParam.iBufferSize = snapshotBufferSize;
getSnapshotParam.iStoreResult = 0;
getSnapshotParam.iNodeNumber = SQLM_CURRENT_NODE;
getSnapshotParam.poOutputFormat = &outputFormat;
getSnapshotParam.iSnapshotClass = SQLM_CLASS_HEALTH;
```

8. ヘルス・スナップショットをキャプチャーします。以下のパラメーターを渡します。
   - db2GetSnapshotData 構造体。これには、スナップショットをキャプチャーするのに必要な情報が含まれています。
   - スナップショット出力の宛先となるバッファーの参照。

```
db2GetSnapshot(db2Version810, &getSnapshotParam, &sqlca);
```

9. バッファーのオーバーフローを処理するためのロジックを組み込みます。スナップショットが取られた後、バッファー・オーバーフローについて sqlcode がチェックされます。バッファー・オーバーフローが発生する場合には、バッファーがクリアされて再初期化され、スナップショットが再度取られます。

```
while (sqlca.sqlcode == 1606)
{
  free(snapshotBuffer);
  snapshotBufferSize += SNAPSHOT_BUFFER_UNIT_SZ;
  snapshotBuffer = (char *)malloc(snapshotBufferSize);
  if (snapshotBuffer == NULL)
  {
    printf("¥nMemory allocation error.¥n");
    return;
  }

  getSnapshotParam.iBufferSize = snapshotBufferSize;
  getSnapshotParam.poBuffer = snapshotBuffer;
  db2GetSnapshot(db2Version810, &getSnapshotParam, &sqlca);
}
```

10. スナップショット・モニターのデータ・ストリームを処理します。スナップショット・モニターのデータ・ストリームを参照するには、これらのステップの後の図を参照してください。
11. バッファーをクリアします。

```
free(snapshotBuffer);
free(pRequestedDataGroups);
```

db2GetSnapshot API でヘルス・スナップショットをキャプチャーした後、ヘルス・スナップショット出力が自己記述型データ・ストリームとして戻されます。以下は、データ・ストリーム構造の例です。

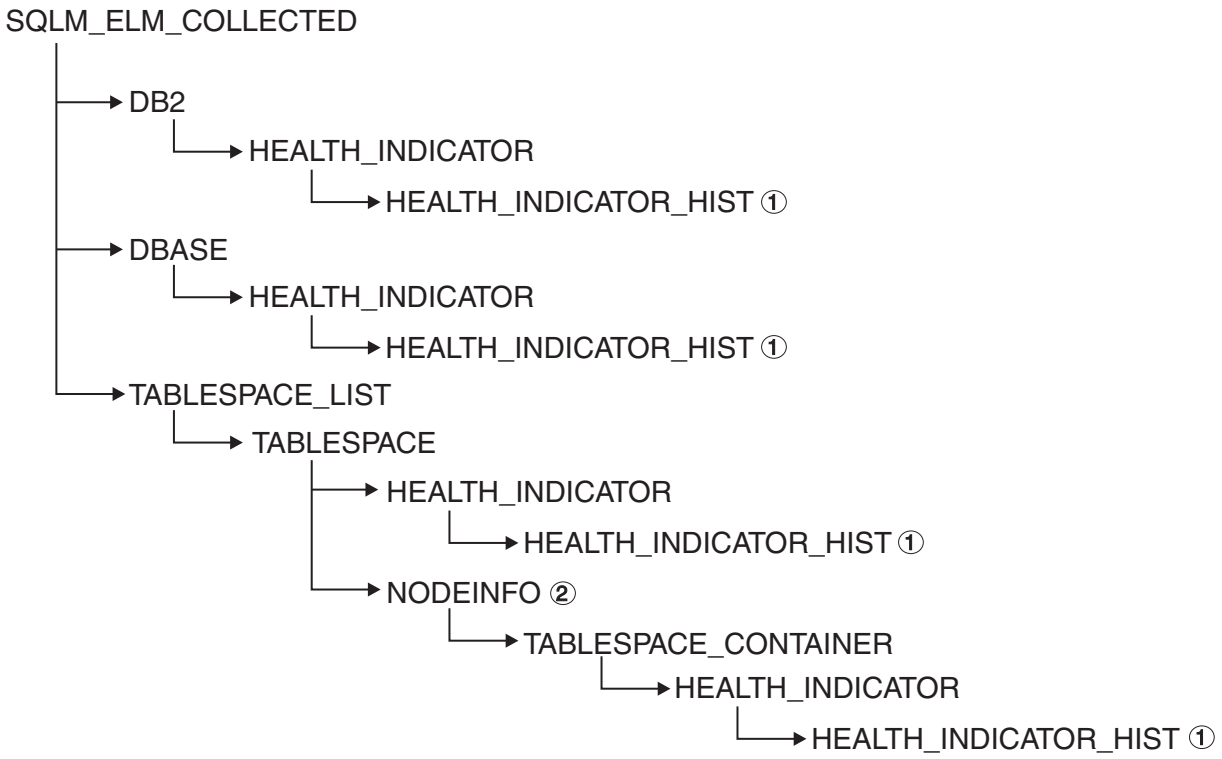


図 5. ヘルス・スナップショット自己記述型データ・ストリーム

凡例:

1. SQLM_CLASS_HEALTH_WITH_DETAIL スナップショット・クラスが使用される場合のみ使用可能。
2. DB2 Enterprise Server Edition でのみ使用可能。それ以外の場合は、表スペース・コンテナー・ストリームになります。

次の階層は、ヘルス・スナップショット自己記述型データ・ストリーム中の特定のエレメントを示しています。

SQLM_ELM_HI の下のエレメントの階層:

```
SQLM_ELM_HI
   SQLM_ELM_HI_ID
   SQLM_ELM_HI_VALUE
      SQLM_ELM_HI_TIMESTAMP
         SQLM_ELM_SECONDS
         SQLM_ELM_MICROSEC
   SQLM_ELM_HI_ALERT_STATE
```

SQLM_ELM_HI_HIST の下のエレメントの階層
(SQLM_CLASS_HEALTH_WITH_DETAIL スナップショット・クラス使用時のみ使用可能):

```
SQLM_ELM_HI_HIST
   SQLM_ELM_HI_FORMULA
   SQLM_ELM_HI_ADDITIONAL_INFO
      SQLM_ELM_HEALTH_INDICATOR_HIST
         SQLM_ELM_HI_ID
         SQLM_ELM_HI_VALUE
            SQLM_ELM_HI_TIMESTAMP
               SQLM_ELM_SECONDS
```

```
            SQLM_ELM_MICROSEC
         SQLM_ELM_HI_ALERT_STATE
         SQLM_ELM_HI_FORMULA
         SQLM_ELM_HI_ADDITIONAL_INFO
```

SQLM_ELM_OBJ_LIST の下のエレメントの階層:

```
SQLM_ELM_HI_OBJ_LIST
   SQLM_ELM_HI_OBJ_NAME
   SQLM_ELM_HI_OBJ_DETAIL
   SQLM_ELM_HI_OBJ_STATE
      SQLM_ELM_HI_TIMESTAMP
         SQLM_ELM_SECONDS
         SQLM_ELM_MICROSEC
```

SQLM_ELM_OBJ_LIST_HIST の下のエレメントの階層 (SQLM_CLASS_HEALTH_WITH_DETAIL スナップショット・クラス使用時のみ使用可能):

```
SQLM_ELM_HI_OBJ_LIST_HIST
   SQLM_ELM_HI_OBJ_NAME
   SQLM_ELM_HI_OBJ_STATE
      SQLM_ELM_HI_TIMESTAMP
         SQLM_ELM_SECONDS
         SQLM_ELM_MICROSEC
```

ヘルス・モニターの出力例:

次の例は、CLP を使用して取ったヘルス・スナップショットとそれらに対応する出力を示し、ヘルス・モニターの性質を図示します。この例の目的は、データベース・マネージャーの開始直後に全体の正常性に関する状況を検査することです。

1. 次のように GET HEALTH SNAPSHOT コマンドを使用して、データベース・マネージャーのスナップショットを取ります。

   ```
   db2 get health snapshot for dbm
   ```

   CLP から GET HEALTH SNAPSHOT コマンドが発行された後、スナップショット出力が画面に送られます。

   ```
   Node name                                    =
   Node type                                    = Database Server with local
                                                  and remote clients
   Instance name                                = DB2
   Snapshot timestamp                           = 11-07-2002 12:43:23.613425

   Number of database partitions in DB2 instance  = 1
   Start Database Manager timestamp             = 11-07-2002 12:43:18.000108
   Instance highest severity alert state        = Not yet evaluated

   Health Indicators:

       Not yet evaluated
   ```

2. この出力を分析します。このヘルス・スナップショットから、インスタンスの最大重大度アラート状態が「Not yet evaluated」であることが分かります。ヘルス・モニターが開始されたばかりで、まだヘルス・インディケーターを評価していないので、インスタンスはこの状態です。

   インスタンスの最大重大度アラート状態が変わらない場合、次のようにします。

   - HEALTH_MON データベース・マネージャー構成パラメーターの値を検査して、ヘルス・モニターが ON かどうか判別する。

- HEALTH_MON=OFF の場合は、ヘルス・モニターが開始されていない。ヘルス・モニターを開始するには、 UPDATE DBM CFG USING HEALTH_MON ON コマンドを発行します。
- HEALTH_MON=ON の場合は、インスタンスにアタッチしてヘルス・モニターを活動化する。インスタンス接続がある場合は、ヘルス・モニターをメモリー中にロードできなかった可能性があります。

CLP を使用してデータベースのヘルス・スナップショットをとる別の例を以下に概略します。

1. 始める前に、データベース接続が存在し、データベースが静止していることを確認してください。
2. 次のように GET HEALTH SNAPSHOT コマンドを使用して、データベース・マネージャーのスナップショットを取ります。

   ```
   db2 get health snapshot for db on sample
   ```

3. CLP から GET HEALTH SNAPSHOT コマンドが発行された後、スナップショット出力が画面に送られます。

   ```
                Database Health Snapshot

   Snapshot timestamp                           = 12-09-2002 11:44:37.793184

   Database name                                = SAMPLE
   Database path                                = E:¥DB2¥NODE0000¥SQL00002¥
   Input database alias                         = SAMPLE
   Operating system running at database server= NT
   Location of the database                     = Local
   Database highest severity alert state        = Attention

   Health Indicators:

    ...
      Indicator Name                               = db.log_util
         Value                                     = 60
         Unit                                      = %
         Evaluation timestamp                      = 12-09-2002 11:44:00.095000
         Alert state                               = Normal

      Indicator Name                               = db.db_op_status
         Value                                     = 2
         Evaluation timestamp                      = 12-09-2002 11:44:00.095000
         Alert state                               = Attention
   ```

4. この出力を分析します。

   このヘルス・スナップショットは、 *db.db_op_status* ヘルス・インディケーター上にアテンション・アラートがあることを示しています。値 2 は、データベースが静止状態であることを示しています。

グローバル・ヘルス・スナップショット:

パーティション・データベース・システムでは、現行パーティション、指定したパーティション、またはすべてのパーティションのヘルス・スナップショットをとることができます。パーティション・データベースのすべてのパーティションに渡ってグローバル・ヘルス・スナップショットをとる場合は、可能であれば、データが集約されてから、結果が戻されます。

ヘルス・インディケーターの集約されたアラート状態は、すべてのデータベース・パーティション間の最大重大度アラート状態と等しくなります。追加情報と履歴データは、データベース・パーティション間で集約できないので使用できません。ヘルス・インディケーターの残りのデータは、以下の表に詳述されているように集約されます。

表 46. ヘルス・インディケーターの値、タイム・スタンプ、および公式データの集約

ヘルス・インディケーター	集約の詳細
• db2.db2_op_status • db2.sort_privmem_util • db2.mon_heap_util • db.db_op_status • db.sort_shrmem_util • db.spilled_sorts • db.log_util • db.log_fs_util • db.locklist_util • db.apps_waiting_locks • db.db_heap_util • db.db_backup_req • ts.ts_util	ヘルス・インディケーターの値は、最大値を含むパーティションから取得される。 評価タイム・スタンプと公式は、同じパーティションから取得される。
• db.max_sort_shrmem_util • db.pkgcache_hitratio • db.catcache_hitratio • db.shrworkspace_hitratio	ヘルス・インディケーターの値は、最小値を含むパーティションから取得される。 評価タイム・スタンプと公式は、同じパーティションから取得される。
• db.deadlock_rate • db.lock_escal_rate	ヘルス・インディケーターの値は、すべてのデータベース・パーティション間の値の合計になる。 評価タイム・スタンプと公式は集約できないので使用できない。
• ts.ts_op_status • tsc.tscont_op_status • tsc.tscont_util	ヘルス・インディケーターは集約されない。
• db.hadr_op_status • db.hadr_log_delay	これらのヘルス・インディケーターは、複数のパーティション・データベースではサポートされない。
• db.tb_reorg_req • db.tb_runstats_req • db.fed_nicknames_op_status • db.fed_servers_op_status	このヘルス・インディケーターは、1 つのパーティションのみに対して評価されるので、集約の必要はない。ヘルス・インディケーターを評価するパーティションからデータが戻されます。

注: 1 つのパーティション・オブジェクトに関するグローバル・スナップショットをとると、集約するパーティションがないので、出力にはすべての属性が含まれます。

ヘルス・モニターのグラフィック・ツール:
ヘルス・センター

ヘルス・センターは、例外による管理をサポートするために設計されたグラフィック管理ツールです。クライアントでカタログされているすべての Windows、Linux、および UNIX のインスタンスとデータベースの場合、ヘルス・センターは以下のものを備えます。

- すべてのインスタンスとそれらのデータベースのロールアップされたアラート状態を表示するセントラル・ロケーション
- インスタンス、データベース、およびそれらの子オブジェクトに関する現行アラートを表示するグラフィカル・インターフェース
- 現行アラートに関する詳細情報と推奨される解決方法にアクセスするグラフィカル・インターフェース

コマンド行からヘルス・センターを開始するには、 `db2hc` コマンドを入力してください。

重要: ヘルス・センターは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。 詳しくは、「*DB2 バージョン 9.7 の新機能*」のトピック『コントロール・センター・ツールおよび DB2 管理サーバー (DAS) が推奨されなくなった』を参照してください。

Windowsでは、「スタート」メニューから、**「スタート」** → **「プログラム」** → **「IBM DB2」** →**<DB2 コピー名>**→ **「モニター・ツール」** → **「ヘルス・センター」** をクリックして、ヘルス・センターを開始することもできます。

ヘルス・センターの左側のパネルにはナビゲーション・ツリーがあり、右側のパネルにはアラート・ビューがあります。ナビゲーション・ビューの内容は、ナビゲーション・ビューの上部で選択したトグル・ボタンに基づいてフィルタリングされます。

ヘルス・センターは、**「任意のアラート状態のオブジェクト (Object in Any Alert State)」**トグル・ボタンが選択された状態で開きます。このボタンは、現行アラートと注意を払う必要のあるインスタンスを関連付けるのに役立ちます。**「すべてのオブジェクト」**トグル・ボタンを選択すると、クライアントでカタログされたすべての Windows、Linux、および UNIX インスタンスと、それぞれの状態が表示されます。アイコンのないインスタンスは、ヘルス・モニターを実行していないか、またはバージョン 8 より前のインスタンス (ヘルス・モニター機能のサポートがない) です。

インスタンスを選択すると、ヘルス・センターは、選択されたインスタンスに関するヘルス・モニターから状況を要求します。アラート・ビューには、インスタンス、そのすべてのデータベース、およびそれらのデータベースごとの表スペースと表スペース・コンテナーに関するすべての現行アラートが表示されます。ナビゲーション・ビュー内のインスタンスを展開して、子データベース・オブジェクトを選択すると、アラート・ビューは、選択されたデータベースとその表スペースまたは表スペース・コンテナーに関するアラートに制限されます。

ヘルス・センターの右上隅にリフレッシュ・アイコンがあります。リフレッシュ・アイコンをクリックして即時リフレッシュするか、特定のリフレッシュ・インター

バルを設定すると、ヘルス・センターは現在の状況についてサーバー上のヘルス・モニターを照会します。この照会により、ヘルス・モニターがヘルス・インディケーター評価をリフレッシュすることはありません。個々のヘルス・インディケーターには、定義済みのリフレッシュ・インターバルがあります。ヘルス・インディケーターがアラート状態に関して再評価されるのは、リフレッシュ・インターバルが経過した場合だけです。ヘルス・センターの時間指定リフレッシュまたは要求リフレッシュのたびに、ヘルス・インディケーターの現在の状況のみ表示されます。

アラート・ビューには、特定のカスタマイズ列やソート順序を含むカスタマイズ・ビューを定義する機能があります。ヘルス・センターには、独自の命名およびカテゴリー化スキームにカスタマイズできる、事前定義済みのビューが 6 つあります。ウィンドウの下部にあるツールバーを使用するか、「**表示**」メニューで「**保管されたビュー**」を選択すると、事前定義済みのビューを選択できます。独自のカスタマイズ・ビューを定義するには、ウィンドウの下部にあるツールバーの「**表示**」ボタンをクリックするか、「**表示**」メニューを使用してください。アラート・ビューでデータを表示するために選択したビューは、次回のヘルス・センターの呼び出し時に記憶されています。

アラートに関する詳細情報を取得するには、アラート・ビューでアラート行を選択してください。「**選択**」メニューを使用するか、行を右クリックして、「**詳細表示**」を選択してください。「詳細」ウィンドウにはアラートに関する詳細情報が表示されます。この情報には、アラートが生じたオブジェクトやパーティション、公式 (該当する場合)、およびヘルス・インディケーターの値が含まれます。

しきい値ベースのヘルス・インディケーターの場合、アラート条件の判別に使用されたしきい値が表示されます。「詳細」ウィンドウにはヘルス・インディケーターに関する追加情報も表示されます。この情報には、構成パラメーターや、アラートのコンテキストを示す他のモニター・データが含まれる場合もあります。ヘルス・インディケーターの目的や測定対象の重要属性である理由を含む、ヘルス・インディケーターの説明が表示されます。

コレクション状態ベースのヘルス・インディケーターの場合、「**ヘルス・インディケーターのアラート状態 (Health Indicator Alert State)**」表の「オブジェクト」に、コレクション・オブジェクトのリストが表示されます。この表には、オブジェクト名、タイム・スタンプ、および詳細情報が表示されます。

詳細ページには「履歴の表示」ボタンがあります。 2 回目のヘルス・インディケーター評価のリフレッシュ以降、ヘルス・インディケーターの履歴レコードが保管されます。履歴レコードの保管後に限り、ヘルス・センター内の「履歴の表示」ダイアログに内容が表示されます。コレクション状態ベースのヘルス・インディケーターの場合、「履歴」ウィンドウ内で「**収集履歴の表示 (View Collection History)**」ボタンをクリックして、収集の履歴を表示できます。

ヘルス・センター状況ビーコン

ヘルス・センター状況ビーコンは、 DB2 管理ツール中で使用可能にできる表示標識です。他の DB2 管理ツールを処理している最中に、ヘルス・センターが開いていないと、このビーコンは現行アラートを通知します。このビーコンは、アラート条件のためにヘルス・センターを開くようユーザーにプロンプトを出すことを意図しています。

ヘルス・センター状況ビーコンには、2 種類の通知方式があります。 1 つ目の通知方式では、ポップアップ・メッセージが使用されます。もう 1 つの通知方式では、オープン・ウィンドウの状況表示行の右側に表示されるグラフィック・ビーコンが使用されます。このグラフィック・ビーコンには、 1 回のクリックでヘルス・センターにアクセスできるボタンが組み込まれています。

両方のビーコン通知方式とも、「ツール設定」ダイアログを使用して使用可能にします。「ポップアップによる通知」方式はポップアップ・メッセージ通知を制御し、「状況表示行による通知」方式は表示ビーコンを制御します。

正常性の推奨事項の取得:

正常性を保つための推奨事項の SQL による照会:

SYSPROC.HEALTH_HI_REC ストアード・プロシージャーを使用して、 SQL で推奨事項を照会できます。

SYSPROC.HEALTH_HI_REC ストアード・プロシージャーを使用すると、推奨事項は次のような XML 文書で戻されます。

- sqllib¥misc ディレクトリー中にある正常性の推奨事項の XML スキーマ DB2RecommendationSchema.xsd に従ってフォーマットされている。
- UTF-8 でエンコードされ、クライアントの言語のテキストが含まれている。
- 推奨事項の集合を収集したものとして編成されている。個々の推奨事項の集合には、解決しようとしている問題 (ヘルス・インディケーター) が記述され、そのヘルス・インディケーターを解決する際の 1 つ以上の推奨事項が含まれます。この文書から検索できる情報に関する特定の詳細情報については、スキーマ定義を参照してください。

CLP を使用して入手できる情報はすべて、 SQL を使用して照会すると戻される XML 推奨文書でも入手できます。

SYSPROC.HEALTH_HI_REC ストアード・プロシージャーは、以下の引数を取ります。

- ヘルス・インディケーター
- ヘルス・インディケーターがアラート状態になっているオブジェクトの定義

出力の推奨文書は BLOB として戻されます。したがって、CLP の場合は表示される出力の量が制限されるので、コマンド行からこのストアード・プロシージャーを処理しても効果的ではありません。戻された XML 文書を適切に解析してご希望のエレメントや属性を検索できる高水準言語 (C や Java など) を使用して、このストアード・プロシージャーを呼び出すことをお勧めします。

正常性を保つための推奨事項の CLP による検索:

CLP から GET RECOMMENDATIONS コマンドを使用して推奨事項を検索できます。このコマンド構文は、推奨事項を照会して、特定のオブジェクト上で現在アラート状態になっているヘルス・インディケーターなどの特定のヘルス・アラートを解決することをサポートしています。

ヘルス・モニターから推奨事項を検索するには、インスタンス接続がなければなりません。インスタンス接続がない場合、デフォルトのインスタンス接続が作成され

ます。リモート・インスタンス上のヘルス・モニターから推奨事項を取得するには、まずそのインスタンスにアタッチする必要があります。ヘルス・モニターから推奨事項を検索するには、特殊権限は必要ありません。

またこのコマンド構文は、特定のヘルス・インディケーターに関する推奨事項の完全集合の検索もサポートしています。このヘルス・インディケーターは、コマンドの実行時にアラート状態になっている必要はありません。特定のヘルス・インディケーターに関するアラートを解決する際の推奨事項は、単一パーティション・レベルかグローバル・レベルのいずれかで照会できます。

特定のオブジェクト上のヘルス・アラートに関する推奨事項を照会すると、ヘルス・モニターは特定のアラートを解決しようとし、出力の問題のセクション中に解決しようとしたアラートに関する詳細情報を示すことができます。

またヘルス・モニターは、推奨事項のランキングを示したり、場合によってはアラートを解決するために実行できるスクリプトを生成することもできます。さらに、一部の推奨事項が特定の問題状態に該当しない場合に、ヘルス・モニターはそれらの推奨事項をリジェクトして非表示にできます。他方、最初の下記の例のように、推奨事項がヘルス・インディケーター名のみで照会されると、考えられる推奨事項の全集合が常に戻されます。この場合、単に CLP コマンドは、ユーザーがアラートを示された場合に考慮する必要のあるアクションに関する情報を示します。

GET RECOMMENDATIONS コマンドを使用して、推奨事項を取得します。

1. 次のコマンドを実行して、 **db.db_op_status** ヘルス・インディケーターに関するアラートを解決する場合に推奨できるアクションの全集合を参照できます。

   ```
   db2 get recommendations for health indicator db.db_op_status
   ```

   この例では、 **db.db_op_status** ヘルス・インディケーターに関する推奨事項の全集合が戻されます。このコマンドを実行する際に、このヘルス・インディケーターはアラート状態になっている必要はありません。

   この出力は、このヘルス・インディケーターに関する推奨事項が 2 つ考えられることを示しています。 1 つはデータベースの活動化で、もう 1 つはデータベースに関するロールフォワード進行状況の調査です。このコマンドを使用すると、特定のアラートの解決方法が要求されるのではなく、考えられる推奨事項がすべて照会されるので、ヘルス・モニターはこの事例の最善の推奨事項を識別できません。その結果、推奨事項の全集合が戻されます。

   ```
   Recommendations:

   Recommendation: Investigate rollforward progress.

   A rollforward is in progress on the database due to an explicit
   request from the administrator. You have to wait for the rollforward
   to complete for the instance to return to active state.

   Take one of the following actions:

   Launch DB2 tool: Utility Status Manager

   The Utility Status Manager allows you to monitor the progress
   and change the priority of currently running utilities.

   To open the Utility Status Manager:
   1.  From the Control Center, expand the object tree until you find
   ```

```
    the database that you want.
2.  Right-click the database, and click Manage Utilities from the
    pop-up menu. The Utility Status Manager opens.

To view progress of the rollforward utility, right-click on the
rollforward utility and select View Progress Details.


From the Command Line Processor, issue the commands shown in the
following example to view the progress of the rollforward
utility:

LIST UTILITIES SHOW DETAIL


Recommendation: Unquiesce the database.

The database has been put into QUIESCE PENDING or QUIESCE state by
an explicit request from the administrator. If you have QUIESCE_CONNECT
authority, or are DBADM or SYSADM, you will still have access to the
database and will be able to use it normally. For all other users, new
connections to the database are not permitted and new units of work
cannot be started. Also, depending on the quiesce request, active
units of work will be allowed to complete or will be rolled back
immediately. You can issue an unquiesce to return to active state.

Take one of the following actions:

Launch DB2 tool: Control Center Unquiesce Database

The Control Center has an option on a database that can be used to
unquiesce the database.

To unquiesce a database:
1.  From the Control Center, expand the object tree until you find
    the database that you want.
2.  Right-click the database, and click Unquiesce from the pop-up
    menu. The database is unquiesced.


From the Command Line Processor, issue the commands shown in the
following example:

CONNECT TO DATABASE database-alias
UNQUIESCE DATABASE
```

2. データベース SAMPLE のヘルス・インディケーター **db.db_heap_util** がアラート状態になっていることに気付き、アラートの解決方法を判別することにしました。この場合、特定の問題を解決することを望んでいるので、次の方法で GET RECOMMENDATIONS コマンドを発行できます。

```
db2 get recommendations for health indicator db.db_heap_util
    for database on sample
```

この出力は、問題のサマリーと、問題を解決するための推奨事項の集合を示しています。ヘルス・モニターは、優先順位に従って推奨事項をランク付けしています。個々の推奨事項には、説明とアクションの集合が含まれていて、推奨アクションを実行する方法が示されています。

```
Problem:

    Indicator Name                          = db.db_heap_util
       Value                                = 42
       Evaluation timestamp                 = 11/25/2003 19:04:54
       Alert state                          = Alarm
```

```
       Additional information                  =

Recommendations:

Recommendation: Increase the database heap size.
Rank: 1

Increase the database configuration parameter dbheap sufficiently
to move utilization to normal operating levels. To increase the
value, set the new value of dbheap to be equal to
(pool_cur_size / (4096*U)) where U is the desired utilization rate.
For example, if your desired utilization rate is 60% of the warning
threshold level, which you have set at 75%, then
U = 0.6 * 0.75 = 0.45 (or 45%).

Take one of the following actions:

Execute the following scripts at the DB2 server (this can be done using
the EXEC_DB2_CMD stored procedure):

CONNECT TO DATABASE SAMPLE;
UPDATE DB CFG USING DBHEAP 149333;
CONNECT_RESET;


Launch DB2 tool: Database Configuration Window

The Database Configuration window can be used to view and update
database configuration parameters.

To open the Database Configuration window:
1.  From the Control Center, expand the object tree until you find
    the databases folder.
2.  Click the databases folder. Any existing database are displayed
    in the contents pane on the right side of the window.
3.  Right-click the database that you want in the contents pane, and
    click Configure Parameters in the pop-up menu. The Database
    Configuration window opens.

On the Performance tab, update the database heap size parameter as
suggested and click OK to apply the update.


Recommendation: Investigate memory usage of database heap.
Rank: 2

There is one database heap per database and the database manager uses
it on behalf of all applications connected to the database. The data
area is expanded as needed up to the maximum specified by dbheap.

For more information on the database heap, refer to the DB2
Information Center.

Investigate the amount of memory that was used for the database heap
over time to determine the most appropriate value for the database
heap configuration parameter. The database system monitor tracks the
highest amount of memory that was used for the database heap.

Take one of the following actions:

Launch DB2 tool: Memory Visualizer

The Memory Visualizer is used to monitor memory allocation within
a DB2 instance. It can be used to monitor overall memory usage, and
to update configuration parameters for individual memory components.

To open the Memory Visualizer:
```

1. From the Control Center, expand the object tree until you find the instances folder.
2. Click the instances folder. Any existing instances are displayed in the contents pane on the right side of the window.
3. Right-click the instance that you want in the contents pane, and click View Memory Usage in the pop-up menu. The Memory Visualizer opens.

To start the Memory Visualizer from the command line issue the db2memvis command.

The Memory Visualizer displays a hierarchical list of memory pools for the database manager. Database Heap is listed under the Database Manager Memory group for each database. On Windows, it is listed under the Database Manager Shared Memory group.

Click the check box on the Show Plot column for the Database Heap row to add the element to the plot.

3. パーティション・データベース・システムの場合、特定のパーティション上のアラート状態になっているヘルス・インディケーターに関する推奨事項を照会することも、すべてのパーティションについてグローバルに照会することもできます。推奨事項をグローバルに照会すると、すべてのパーティション上のヘルス・インディケーターに適用される推奨事項の集合が戻されます。例えば、パーティション 1 と 3 上でヘルス・インディケーターがアラート状態になっている場合は、 2 つのスクリプトを収集したものが戻され、それぞれのスクリプトは別のパーティションに適用されます。

   次の例は、特定のパーティション (この例ではパーティション番号 2) 上のヘルス・インディケーターに関する推奨事項を照会する方法を示しています。

```
db2 get recommendations for health indicator db.db_heap_util
  for database on sample at dbpartitionnum 2
```

   次の例は、複数のパーティション上でアラート状態になっているヘルス・インディケーターを解決するための推奨事項の集合を検索する方法を示しています。

```
db2 get recommendations for health indicator db.db_heap_util
  for database on sample global
```

システムの正常性を保つための推奨事項をクライアント・アプリケーションを使用して検索:

C または C++ アプリケーションで db2GetRecommendations API を使用して、推奨事項を照会できます。

ヘルス・スナップショットをキャプチャーするには、インスタンス接続がなければなりません。インスタンス接続がない場合、デフォルトのインスタンス接続が作成されます。リモート・インスタンスに関する推奨事項を照会するには、まずそのインスタンスにアタッチする必要があります。

db2GetRecommendations API を使用すると、推奨事項は次のような XML 文書で戻されます。

- SQLLIB ディレクトリー中の MISC サブディレクトリー中にある正常性の推奨事項の XML スキーマ DB2RecommendationSchema.xsd に従ってフォーマットされている。
- UTF-8 でエンコードされ、クライアントの言語のテキストが含まれている。

- 推奨事項の集合を収集したものとして編成されている。個々の推奨事項の集合には、解決しようとしている問題 (ヘルス・インディケーター) が記述され、そのヘルス・インディケーターを解決する際の 1 つ以上の推奨事項が含まれます。この文書から検索できる情報に関する特定の詳細情報については、スキーマ定義を参照してください。

CLP を使用して入手できる情報はすべて、戻される XML 推奨文書でも入手できます。

クライアント・アプリケーションを使用して正常性の推奨事項を検索するには、次のようにします。

1. sqlmon.h および db2ApiDf.h DB2 ヘッダー・ファイルを組み込みます。これらのファイルは、 sqllib¥include ディレクトリーにあります。

   ```
   #include <db2ApiDf.h>
   #include <sqlmon.h>
   ```

2. sqlca および db2GetRecommendationsData 構造体を宣言します。

   ```
   struct sqlca sqlca ;
   db2GetRecommendationsData recData ;

   memset( &sqlca, '¥0', sizeof( struct sqlca ) ) ;
   memset( &recData, '¥0', sizeof( db2GetRecommendationsData ) ) ;
   ```

3. 推奨事項を検索するアラートに関する情報を、 db2GetRecommendationsData 構造体に取り込みます。次のコードの抜粋では、 Sample データベース上の **db2.db_heap_util** ヘルス・インディケーターに関する推奨事項が照会されます。

   ```
   recData.iSchemaVersion = DB2HEALTH_RECSCHEMA_VERSION8_2 ;
   recData.iNodeNumber = SQLM_CURRENT_NODE ;
   recData.iIndicatorID = SQLM_HI_DATABASE_HEAP_UTILIZATION  ;
   recData.iObjType = DB2HEALTH_OBJTYPE_DATABASE ;
   recData.piDbName = "SAMPLE" ;
   ```

4. db2GetRecommendations API を呼び出して、指定データベース上のこのヘルス・インディケーターのアラートに関する推奨事項を検索します。

   ```
   db2GetRecommendations(  db2Version820, &recData, &sqlca ) ;
   ```

5. sqlca 中に戻された sqlcode を検査して、発生したエラーをチェックします。 API 呼び出しが正常に実行された場合は、 db2GetRecommendationsData 構造体の poRecommendation フィールド中に戻された推奨事項の XML 文書を処理してください。 XML パーサーの選択項目を使用して、ご希望のエレメントまたは属性を抽出してください。 XML 文書から取り出せる情報に関する詳細は、 sqllib¥misc ディレクトリー中の DB2RecommendationSchema.xsd XML スキーマを参照してください。

6. db2GetRecommendations API によって割り振られたメモリーを解放します。こうすると、 db2GetRecommendationsData 構造体の poRecommendation フィールド中に戻された推奨事項の文書が解放されます。

   ```
   db2GetRecommendationsFree( db2Version820, &recData, &sqlca );
   ```

普通は、ヘルス・インディケーターがアラート状態になっているのを検出した時点で推奨事項を照会するので、通常は上記のコードとスナップショット API に対する呼び出しを組み合わせて、ヘルス・スナップショットを取ります。

ヘルス・センターを使用したヘルス・モニター・アラートの解決:

ヘルス・センターには、アラート条件に関する推奨アクションを検索してインプリメントするサポートが備えられています。

ヘルス・センターを使用してヘルス・モニター・アラートを解決するには、以下のようにします。

1. ヘルス・センターの「アラート」ビューで、解決するアラートの行を右クリックして、ポップアップ・メニューから「推奨アドバイザー」を選択します。推奨アドバイザーが開いて、「詳細」ウィンドウと同様の形式でアラートの詳細が表示されます。
2. 推奨アドバイザーのステップに従って、最適な推奨事項を選択します。推奨アドバイザーには、推奨事項をインプリメントする機能が備えられています。

推奨事項には、調査と推奨事項という 2 つのタイプがあります。推奨アドバイザーには、これらの推奨事項のタイプに関する次の 4 種類のアクションがサポートされています。

グラフィック管理ツールの立ち上げ
このオプションは、アラート条件の解決や調査を行うグラフィック・ツールを立ち上げます。このツールは、アラートが発生したオブジェクトのコンテキスト中で立ち上げられます。

構成パラメーターの更新
更新する必要のある構成パラメーターと、現行値および推奨値がリストされます。必要に応じて、推奨値を更新できます。

DB2 コマンド・スクリプトの実行
推奨アクションには、複数のコマンドが必要になる場合があります。 DB2 コマンド・スクリプトを使用すると、複数のコマンドを実行して、アラート条件を解決できます。例えば、再編成の必要性ヘルス・インディケーターには、ユーティリティーを実行する DB2 コマンド・スクリプト・アクションが備えられています。

代替解決方法のインプリメント
DB2 管理ツール・セットでアクションを実行できない場合に、代替方式を使用してアラート条件を解決するための指示が備えられています。

ヘルス・インディケーターの構成: デフォルトのヘルス・モニター構成は、インストール時に備えられます。したがって、DB2 の開始直後に、ヘルス・モニターがデータベース環境の正常性を評価できることが保証されます。しかし、特定のユーザーの環境用の構成を使用して、ヘルス・モニターがヘルス・インディケーターを評価したりアラート状態に対応したりする動作を微調整できます。

構成を定義できるレベルは複数あります。 DB2 のインストール時に、ヘルス・インディケーターごとに工場出荷時設定のデフォルト構成が備えられます。初めてヘルス・モニターを開始する際に、工場出荷時設定のコピーにより、デフォルトのインスタンス設定とグローバル設定が備えられます。

インスタンス設定は、インスタンスに適用されます。グローバル設定は、インスタンス中のカスタマイズ設定が定義されていないデータベース、表スペース、および表スペース・コンテナーなどのオブジェクトに適用されます。

特定のデータベース、表スペース、または表スペース・コンテナーに関するヘルス・インディケーター設定を更新すると、更新されたヘルス・インディケーターのオブジェクト設定が作成されます。オブジェクト設定のデフォルトは、グローバル設定です。

ヘルス・モニターは、特定のデータベース、表スペース、表スペース・コンテナーに関するヘルス・インディケーターを処理する際に、オブジェクト設定を検査します。特定のヘルス・インディケーターの設定が一度も更新されていない場合は、デフォルトのグローバル設定を使用してヘルス・インディケーターが処理されます。ヘルス・モニターがインスタンスに関するヘルス・インディケーターを処理する際には、インスタンス設定が使用されます。

ヘルス・インディケーターごとに構成できる複数の属性を使用して、ヘルス・モニターの動作を変更できます。最初のパラメーターの集合 (評価フラグ、しきい値、感度) は、ヘルス・モニターがヘルス・インディケーターに関するアラートを生成する時点を定義します。 2 番目のパラメーターの集合 (アクション・フラグ、アクション) は、アラート生成時のヘルス・モニターの実行内容を定義します。

評価フラグ

個々のヘルス・インディケーターには、アラート状態の評価を使用可能にしたり使用不可にしたりする評価フラグがあります。

警告およびアラームのしきい値

しきい値ベースのヘルス・インディケーターには、ヘルス・インディケーター値の警告およびアラーム領域を定義する設定があります。特定のデータベース環境に合わせて、警告およびアラームのしきい値に変更を加えることができます。

感度パラメーター

感度パラメーターは、アラートが生成される前に、ヘルス・インディケーター値がアラート状態になっていなければならない期間の最小値を秒単位で定義します。感度値に関連した待機時間は、ヘルス・インディケーター値がアラート状態になった最初のリフレッシュ・インターバルの際に開始されます。この値を使用すると、リソースが一時的に使用できないだけでアラートが誤って生成されてしまうことを防止できます。

ログ使用率 (*db.log_util*) ヘルス・インディケーターを使用する例を考えてみましょう。週単位で DB2 通知ログを検討するとします。第 1 週に、*db.log_util* の項目はアラーム状態になっています。この状態に関する通知を受け取ったことを思い出しましたが、 CLP からアラート状態を検査すると、ヘルス・インディケーターは正常な状態に戻っていました。第 2 週の後に、週の同じ時点で同じヘルス・インディケーターに関する 2 度目のアラーム通知項目を受け取りました。アラートが生成された 2 つの事例に関してデータベース環境のアクティビティーを調査して、コミットに長時間を要するアプリケーションが毎週実行されていたことが判明しました。このアプリケーションは、アプリケーションがコミットするまでの短時間 (約 8 分から 9 分)、 ログ使用率の急上昇の原因となっています。通知ログ中のアラーム通知レコード中の履歴項目から、 *db.log_util* ヘルス・インディケーターが 10 分ごとに評価されていたことを判別できます。アラートが生成されているので、アプリケーション時間はこのリフレッシュ・インターバルをまたいでいるはずです。そこで、 *db.log_util* パラメーターの感度を 10

分に設定します。これで、 *db.log_util* の値が初めて警告またはアラームのしきい値の領域に入るたびに、アラートが生成されるにはその前にこの値が 10 分以上その領域に入り続けていなければなりません。アプリケーション時間は 8、9 分のみなので、この状態に関する通知項目が通知ログに記録されることはなくなります。

アクション・フラグ
アラート生成に関するアクションの実行は、アクション・フラグによって制御されます。アクション・フラグが使用可能な場合のみ、構成済みのアラートが実行されます。

アクション
スクリプト・アクションかタスク・アクションを構成して、アラートの発生時に実行できます。しきい値ベースのヘルス・インディケーターの場合、警告またはアラームのしきい値に応じて実行するようにアクションを構成できます。状態ベースのヘルス・インディケーターの場合、通常以外のすべての条件に応じて実行するようにアクションを構成できます。アクションを実行するには、DB2 Administration Server を実行していなければなりません。

次の入力パラメーターが、すべてのオペレーティング・システムのコマンド・スクリプトに渡されます。

- <health indicator shortname>
- <object name>
- <value | state>
- <alert type>

スクリプト・アクションは、オペレーティング・システム上のデフォルトのインタープリターを使用します。デフォルト以外のインタープリターを使用する場合は、タスク・センターでスクリプトの内容を使用してタスクを作成してください。複数のパーティションがある環境では、スクリプト・アクション中に定義されたスクリプトは、すべてのパーティションからアクセス可能でなければなりません。

ヘルス・モニターが個々のヘルス・インディケーターを検査するリフレッシュ・インターバルは構成できません。ヘルス・モニターによって考慮される推奨アクションも構成できません。

ヘルス・モニターの構成は、バイナリー・ファイル HealthRules.reg に保管されます。

- Windows では、HealthRules.reg は x:¥<SQLLIB_PATH>¥<INSTANCE_NAME> に保管されます。例えば d:¥sqllib¥DB2 のようになります。
- UNIX では、 HealthRules.reg は ~/<SQLLIB_PATH>/cfg 中に保管される。例えば ~/home/sqllib/cfg のようになります。

ヘルス・モニターの構成を、Linux、UNIX、または Windows サーバー上の他の DB2 バージョン 8 インスタンスに複製できます。このレプリケーションを行うには、このバイナリー構成ファイルを、ターゲット・インスタンス上の該当するディレクトリーにコピーします。

CLP を使用したヘルス・インディケーター構成の検索:

GET ALERT CONFIGURATION コマンドを使用すると、工場出荷時設定、インスタンス設定、グローバル設定、およびオブジェクト設定を表示できます。

1. データベース・レベルのヘルス・インディケーターのグローバル設定を表示するには、次のコマンドを発行します。この設定は、ヘルス・インディケーターのカスタマイズ設定のないすべてのデータベースに適用されます。

   ```
   DB2 GET ALERT CONFIGURATION FOR DATABASES
   ```

2. データベース・レベルのヘルス・インディケーターのグローバル設定を表示するには、次のコマンドを発行します。この設定は、ヘルス・インディケーターのカスタマイズ設定のないすべてのデータベースに適用されます。

   ```
   DB2 GET ALERT CONFIGURATION FOR DATABASES
   ```

   個々のヘルス・インディケーターの設定の出力は、デフォルトから変更されたかどうかを示します。以下の出力では、グローバル設定は更新されていません。したがって、デフォルトの工場出荷時設定と同じです。データベース・レベルのヘルス・インディケーターの工場出荷時設定を表示するには、上記の例と同じコマンドに DEFAULT キーワードを指定して発行してください。

   ```
            Alert Configuration

   Indicator Name                    = db.db_op_status
       Default                       = Yes
       Type                          = State-based
       Sensitivity                   = 0
       Formula                       = db.db_status;
       Actions                       = Disabled
       Threshold or State checking   = Enabled

   Indicator Name                    = db.sort_shrmem_util
       Default                       = Yes
       Type                          = Threshold-based
       Warning                       = 70
       Alarm                         = 85
       Unit                          = %
       Sensitivity                   = 0
       Formula                       = ((db.sort_shrheap_allocated/sheapthres_shr)
                                        *100);
       Actions                       = Disabled
       Threshold or State checking   = Enabled
   ...
   ```

3. SAMPLE データベースのカスタム設定を表示するには、次のコマンドを発行してください。

   ```
   DB2 GET ALERT CONFIGURATION FOR DATABASE ON SAMPLE
   ```

   指定したオブジェクト上に特定のヘルス・インディケーターに関する固有の設定がない場合は、すべてのデータベースのグローバル設定が表示されます。特定のヘルス・インディケーターの設定を表示するには、前述の例に USING *health-indicator-name* 節を追加してください。

CLP を使用したヘルス・インディケーター構成の更新: 特定のヘルス・インディケーターに関するヘルス・インディケーター構成中で、グローバル設定または特定のオブジェクトのオブジェクト設定を更新できます。

UPDATE ALERT CONFIGURATION コマンドには、さまざまな更新オプションに対応した 4 つの副節があります。個々の UPDATE ALERT CONFIGURATION コマンド中で使用できる副節は 1 つのみです。複数のオプションを使用するには、複数の UPDATE ALERT CONFIGURATION コマンドを発行しなければなりません。

1 つ目の副節 SET *parameter-name value* は、次のものの更新をサポートしています。

- 評価フラグ
- 警告およびアラームしきい値 (該当する場合)
- 感度フラグ
- アクション・フラグ

これらの設定のパラメーター名は、それぞれ次のとおりです。

- THRESHOLDSCHECKED
- WARNING および ALARM
- SENSITIVITY
- ACTIONSENABLED

他の 3 つの副節は、スクリプト・アクションやタスク・アクションの追加、更新、および削除をサポートしています。

次のコマンドは、 SAMPLE データベース上の *db.spilled_sorts* ヘルス・インディケーターに関するしきい値ベースのヘルス・インディケーター構成を更新します。この更新により、警告しきい値が 25 に変更され、アクションが使用可能になり、スクリプト・アクションが追加されます。

```
DB2 UPDATE ALERT CONFIGURATION FOR DATABASE ON SAMPLE USING DB.SPILLED_SORTS
    SET WARNING 25, ACTIONSENABLED YES
DB2 UPDATE ALERT CONFIGURATION FOR DATABASE ON SAMPLE USING DB.SPILLED_SORTS
    ADD ACTION SCRIPT c:¥myscript TYPE OS COMMAND LINE PARAMETERS 'space'
    WORKING DIRECTORY c:¥ ON ALARM USER dba1 PASSWORD dba1
```

次のコマンドは、 *ts.ts_util* ヘルス・インディケーターに関する状態ベースのヘルス・インディケーター構成中のグローバル設定を更新します。この更新により、表スペースがバックアップ・ペンディング状態にある際に実行するアクションが定義されます。

```
DB2 UPDATE ALERT CONFIGURATION FOR TABLESPACES USING TS.TS_UTIL
    SET ACTIONSENABLED YES
DB2 UPDATE ALERT CONFIGURATION FOR TABLESPACES USING TS.TS_UTIL
    ADD ACTION TASK 0.1 ON ATTENTION 32 ON localhost USER dba1 PASSWORD dba1
```

この更新は、このヘルス・インディケーターのカスタマイズ設定のないインスタンスの表スペースすべてに適用されます。

ヘルス・インディケーター構成にアクションを追加する場合、 ON *condition* 節のオプションは、ヘルス・インディケーターのタイプに基づいて異なります。

- しきい値ベースのヘルス・インディケーターの場合、 WARNING および ALARM が有効な条件。
- 状態ベースのヘルス・インディケーターの場合、 ON ATTENTION *state* オプションを使用する必要があります。定義済みの、ヘルス・インディケーターにとって有効な数値の状態を使用する必要があります。データベース・マネージャーとデータベースの操作可能状態の値は、sqllib¥include¥sqlmon.h 中にあります。表スペースと表スペース・コンテナーの操作可能値は、sqllib¥include¥sqlutil.h 中にリストされています。データベース・マネージャーが停止状態にある場合にはアクションを実行することができません。詳しくは、db2.db2_op_status ヘルス・インディケーターの説明を参照してください。

CLP を使用したヘルス・インディケーター構成のリセット:

CLP は、グローバル設定を工場出荷時設定にリセットすることをサポートしています。特定のオブジェクトのオブジェクト設定を、そのオブジェクト・タイプのカスタム設定にリセットすることもできます。

- SAMPLE データベースのオブジェクト設定をデータベースの現行のグローバル設定にリセットするには、以下のようにします。

  ```
  DB2 RESET ALERT CONFIGURATION FOR DATABASE ON SAMPLE
  ```

- データベースのグローバル設定を工場出荷時設定にリセットするには、次のコマンドを発行してください。

  ```
  DB2 RESET ALERT CONFIGURATION FOR DATABASES
  ```

- 特定のヘルス・インディケーターの構成をリセットするには、前述の例に `USING health-indicator-name` 節を追加してください。

クライアント・アプリケーションを使用したヘルス・インディケーターの構成:

ヘルス・モニターの構成は、C または C++ アプリケーション中の db2GetAlertCfg、 db2UpdateAlertCfg、および db2ResetAlertCfg API によってアクセスできます。これらの各 API は、工場出荷時設定、インスタンス設定、グローバル設定、およびオブジェクト設定にアクセスできます。

ヘルス・モニター構成にアクセスするには、インスタンス接続がなければなりません。インスタンス接続がない場合、デフォルトのインスタンス接続が作成されます。リモート・インスタンスのヘルス・モニター構成にアクセスするには、まずそのインスタンスにアタッチする必要があります。

db2GetAlertCfgData 構造中の **objType** パラメーターと **defaultType** パラメーターを組み合わせて使用すると、さまざまなレベルのヘルス・インディケーター構成にアクセスできます。

表 47. objType および defaultType が構成レベルにアクセスする際の設定

設定	objType および defaultType
工場出荷時設定	objType = DB2ALERTCFG_OBJTYPE_{DBM | DATABASES |TABLESPACES | CONTAINERS} および defaultType = DB2ALERTCFG_DEFAULT
グローバル設定	objType = DB2ALERTCFG_OBJTYPE_{DBM | DATABASES | TABLESPACES | CONTAINERS} and defaultType = DB2ALERTCFG_NOT_DEFAULT または objType = DB2ALERTCFG_OBJTYPE_{DATABASE | TABLESPACE | CONTAINER} および defaultType = DB2ALERTCFG_DEFAULT
オブジェクト設定	objType = DB2ALERTCFG_OBJTYPE_{DATABASE | TABLESPACE | CONTAINER} および defaultType = DB2ALERTCFG_NOT_DEFAULT

1. SAMPLE データベース上のヘルス・インディケーターに関する特定のオブジェクト設定を取得するには、次のようにします。

a. sqllib¥include ディレクトリー中にある db2ApiDf.h DB2 ヘッダー・ファイルを組み込みます。

```
#include <db2ApiDf.h>
```

b. sqlca および db2GetAlertCfgData 構造体を宣言して初期化します。

```
struct sqlca ca;
memset (&sqlca, '¥0', sizeof(struct sqlca));

char* objName = NULL;
char* dbName = "SAMPLE";
db2Uint32 objType = DB2ALERTCFG_OBJTYPE_DATABASE;
db2Uint32 defaultType = DB2ALERTCFG_NOT_DEFAULT;

db2GetAlertCfgData data = {objType, objName, defaultType, dbName, 0, NULL} ;
```

c. db2GetAlertCfg API を呼び出します。

```
rc = db2GetAlertCfg (db2Version810, &data, &ca);
```

d. 戻された構成を処理し、API によって割り当てられたバッファーを解放します。

```
if (rc >= SQLO_OK) {
  if ((data.ioNumIndicators > 0) && (data.pioIndicators != NULL)) {
    db2GetAlertCfgInd *pIndicators = data.pioIndicators;

    for (db2Uint32 i=0; i  data.ioNumIndicators; i++) {
 //process the entry as necessary using fields defined in db2ApiDf.h
    }
  }

  db2GetAlertCfgFree (db2Version810, &data, &ca);
}
```

2. 次のステップは、 **db.sort_shrmem_util** ヘルス・インディケーターのアラート構成中で、データベース・オブジェクトのグローバル設定を更新し、警告しきい値を 80 に設定してタスク・アクション 1.1 を追加する手順を詳述しています。

a. sqllib¥include ディレクトリー中にある db2ApiDf.h DB2 ヘッダー・ファイルを組み込みます。

```
#include <db2ApiDf.h>
```

b. sqlca および db2AlertTaskAction 構造体を宣言して初期化します。

```
struct sqlca ca;
memset (&sqlca, '¥0', sizeof(struct sqlca));

db2Uint32 objType = DB2ALERTCFG_OBJTYPE_DATABASES;

db2Uint32 taskCondition = DB2ALERTCFG_CONDITION_WARNING;
char* taskname = "1.1";
char* hostname = NULL;
char* userid = "nobody";
char* password = "nothing";

db2AlertTaskAction newTask={taskname,taskCondition,userid,password,hostname};
```

c. db2UpdateAlertCfgData 構造体を宣言して初期化します。

```
struct db2UpdateAlertCfgData setData;

setData.iObjType = objType;
setData.piObjName = NULL;
setData.piDbName = NULL;

setData.iIndicatorID = 1002;

setData.iNumIndAttribUpdates = 1;
setData.piIndAttribUpdates[0].iAttribID = DB2ALERTCFG_WARNING;
setData.piIndAttribUpdates[0].piAttribValue == 80;

setData.iNumActionUpdates = 0;
setData.piActionUpdates = NULL;
```

```
setData.iNumActionDeletes = 0;
setData.piActionDeletes = NULL;

setData.iNumNewActions = 1;
setData.piNewActions[0].iActionType = DB2ALERTCFG_ACTIONTYPE_TASK;
setData.piNewActions[0].piScriptAttribs = NULL;
setData.piNewActions[0].piTaskAttribs = &newTask;
```

d. db2UpdateAlertCfg API を呼び出します。

```
rc = db2UpdateAlertCfg(db2Version810, &setData, &ca);
```

3. 次のステップは、 SAMPLE データベース中の MYTS 表スペースのカスタム設定をリセットする手順を詳述しています。
   a. sqllib¥include ディレクトリー中にある db2ApiDf.h DB2 ヘッダー・ファイルを組み込みます。

```
#include <db2ApiDf.h>
```

   b. sqlca および db2ResetAlertCfgData 構造体を宣言して初期化します。

```
struct sqlca ca;
memset (&sqlca, '¥0', sizeof(struct sqlca));

char* objName = "MYTS";
char* dbName = "SAMPLE";
db2Uint32 objType = DB2ALERTCFG_OBJTYPE_TABLESPACE;

db2ResetAlertCfgData data = {objType, objName, dbName};
```

   c. db2ResetAlertCfg を呼び出します。

```
rc = db2ResetAlertCfg (db2Version810, &data, &ca);
```

ヘルス・センターを使用したヘルス・インディケーターの構成:

ヘルス・センターには、ヘルス・インディケーターの構成の表示、更新、およびリセットを行うグラフィカル・インターフェースが備えられています。ヘルス・インディケーターの構成は、インスタンス中のヘルス・モニター中に保管されます。

ヘルス・インディケーターのしきい値または感度の設定値を定義、変更、使用可能化、または使用不可化したり、ヘルス・インディケーターにヘルス・アラートが発生した場合の実行中のタスクとスクリプトの定義、変更、使用可能化、または使用不可化したりするには、以下のいずれかの権限がなければなりません。

- SYSADM
- SYSMAINT
- SYSCTRL

インスタンスのヘルス・インディケーター設定値、インスタンスに含まれているデータベース・オブジェクトのグローバル・ヘルス・インディケーター設定値、および個々のデータベース・オブジェクトのグローバル・ヘルス・インディケーター設定値を調整できます。

1. ヘルス・センターを使用してヘルス・インディケーターを構成するには、以下のようにします。
   a. 構成するヘルス・インディケーターのインスタンスを選択します。
   b. 「**選択**」メニューまたは右クリック・メニューから、「**構成**」をクリックしてから、「**ヘルス・インディケーターの設定**」をクリックします。「ヘルス・インディケーター構成ランチパッド」が開きます。

c. このランチパッドには、更新できる構成設定のレベルごとにボタンがあります。表示、更新、またはリセットを行いたい構成のレベルに関するボタンを選択します。個々のボタンにより、選択した構成設定レベルの「ヘルス・インディケーターの構成」ウィンドウが立ち上げられます。

d. ヘルス・インディケーターの設定を更新するには、「現在のヘルス・インディケーター設定」表のヘルス・インディケーターの行を選択します。

e. 「**選択**」メニューまたは右クリック・メニューから、「**編集**」を選択します。 「ヘルス・インディケーターの構成」ノートブックがオープンし、次の情報が表示されます。

- 「**詳細情報**」をクリックすると、ヘルス・インディケーターの説明が表示される。
- 「**評価**」チェック・ボックスを使用することにより、ヘルス・インディケーターの評価を有効にしたり、無効にしたりできる。

  注: 現行アラートに関する右クリック・メニュー・オプションを使用して、ヘルス・センターの「アラート」ビューから現行アラートの「**評価**」フラグを使用不可にすることもできます。このオプションは、次回ヘルス・モニター中のインディケーターをリフレッシュする際に、ヘルス・インディケーターの評価を使用不可にします。ヘルス・センター内でアラートに関する「**評価を使用不可にする**」を選択すると、ヘルス・インディケーターに関する評価フラグは false に設定されますが、次のイベントが起きるまで「アラート」ビューからアラートは除去されません。
  - この特定のヘルス・インディケーターのヘルス・モニター・リフレッシュ・インターバルに達する。
  - ヘルス・モニターがヘルス・インディケーター評価をリフレッシュする。
  - ヘルス・センターが状況の表示をリフレッシュする。
- しきい値ベースのヘルス・インディケーターの場合、「アラート」ページで、警告とアラームのしきい値を更新できる。ヘルス・インディケーターの感度もこのページで設定できます。
- 「アクション」ページで、アラート発生時に実行するタスクまたはスクリプト・アクションを選択できる。しきい値ベースのヘルス・インディケーターの場合は警告またはアラーム条件に応じて実行し、状態ベースのヘルス・インディケーターの場合は正常以外の条件に応じて実行するようにアクションを構成できます。「**アクションを有効にする**」チェック・ボックスを選択したり選択解除したりして、アクションの実行を使用可能にしたり使用不可にしたりできます。タスクまたはスクリプト・アクションの追加、更新、または除去を行うには、「**スクリプト・アクション**」および「**タスク・アクション**」表の隣のボタンを使用してください。

2. インスタンスの工場出荷時ヘルス・インディケーター設定を表示するには、次のようにします。

a. 「ヘルス・インディケーターの構成」ランチパッドで、「**インスタンス設定**」をクリックします。

b. 「インスタンス・ヘルス・インディケーターの構成」ウィンドウで、「**デフォルトの表示**」をクリックします。

3. データベース、表スペース、または表スペース・コンテナーのグローバル・ヘルス・インディケーター設定を表示するには、次のようにします。
   a. 「ヘルス・インディケーターの構成」ランチパッドで、「**グローバル設定**」をクリックします。
   b. 「グローバル・ヘルス・インディケーターの構成」ウィンドウで、オブジェクト・タイプを選択します。
   c. これらのグローバル設定の出荷時のデフォルト値を表示するには、「**デフォルトの表示**」をクリックします。
4. データベース・オブジェクトのヘルス・インディケーター設定を表示するには、次のようにします。
   a. 「ヘルス・インディケーターの構成」ランチパッドで、「**オブジェクト設定**」をクリックします。
   b. 「オブジェクト・ヘルス・インディケーターの構成」ウィンドウで、オブジェクトを選択します。
   c. このオブジェクト・タイプのグローバル・ヘルス・インディケーター設定のデフォルトを表示するには、「**デフォルトの表示**」をクリックします。

これらの各ウィンドウで、表示されているすべてのヘルス・インディケーターの設定をデフォルトにリセットするには、「**デフォルトにリセット**」をクリックします。「**現在のヘルス・インディケーター設定**」フィールドでヘルス・インディケーターを 1 つ以上右クリックして、ポップアップ・メニューから「**デフォルトにリセット**」を選択することにより、個々のヘルス・インディケーターをリセットすることもできます。

組み合わせの状態に対するヘルス・モニターのアラート・アクション:

アラート・アクションは、ヘルス・インディケーターがアラート状態に入るときに実行されるタスクまたはスクリプトです。

DB2 V9.1 以降、ヘルス・インディケーター **ts.ts_op_status** に定義されるヘルス・モニターのアラート・アクションが 1 つのアラート状態にある場合、その他の組み合わせの状態に関係なく、表スペースにこの状態が設定されるといつでも実行されます。これにより、他の状態と一緒に設定されていても、特定の表スペースの状態に対してアラート・アクションを実行することが可能になります。

以下の例の場合、アテンション状態 QUIESCED:share に対して定義されているアラート・アクション script1 は、表スペース状態が同時に QUIESCED:share と QUIESCE:update になる場合でも実行されます。

```
db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
    set actionsenabled yes')

db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
    add action script /home/guest001/script1 type operating system command line
    parameters userParam working directory /home/guest001/
    on attention QUIESCED_SHARE on aix1 user guest001 using passw0rd')
```

以下の例の場合、状態の組み合わせ (QUIESCED:share + QUIESCED:update = 3) によって定義されているアラート・アクションは、表スペースの状態が QUIESCED:share と QUIESCED:update の両方になっている場合に限って実行されます。

```
db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
    set actionsenabled yes')

db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
    add action script /home/guest001/script1 type operating system command line
    parameters userParam working directory /home/guest001/
    on attention 3 on aix1 user guest001 using passw0rd')
```

DB2 V9.1 以降、同じアクション属性 (名前、作業ディレクトリー、コマンド行パラメーター、ホスト、ユーザーおよびパスワード) を持つオブジェクトに対して定義されたヘルス・モニターのアラート・アクションは、複数のアラート状態に対して定義された場合でも一度しか実行されません。

以下の例では、2 つの別々のアラート状態に対して同じアクションが定義されています。このアクションは、表スペース状態が QUIESCED:share と QUIESCED:update の両方になっている場合でも、特定の表スペースに対して一度しか実行されません。

```
db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
    add action script /home/guest001/script1 type operating system command line
    parameters userParam working directory /home/guest001/
    on attention QUIESCED_SHARE on aix1 user guest001 using passw0rd')

db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
    add action script /home/guest001/script1 type operating system command line
    parameters userParam working directory /home/guest001/
    on attention QUIESCED_UPDATE on aix1 user guest001 using passw0rd')
```

ヘルス・インディケーター

ヘルス・モニターは、ヘルス・インディケーターを使用して、データベース・マネージャーやデータベースのパフォーマンスの特定の性質の正常性を評価します。ヘルス・インディケーターは、表スペースなど特定のクラスのデータベース・オブジェクトのある性質の正常性を測定します。正常性を判別するため、測定値に対してある基準が適用されます。ここで適用される基準は、ヘルス・インディケーターのタイプに従属します。この基準に基づいて正常稼働ではないと判別されると、アラートが生成されます。

ヘルス・モニターによって、以下の 3 つのタイプのヘルス・インディケーターが戻されます。

- **しきい値ベースの**インディケーターは、オブジェクトの動作の (連続的な範囲の値の) 統計を表すメジャーである。警告およびアラームしきい値は、正常、警告、およびアラーム範囲の境界つまりゾーンを定義します。しきい値ベースのヘルス・インディケーターには、正常、警告、およびアラームの 3 つの有効な状態があります。
- **状態ベースの**インディケーターは、データベース・オブジェクトまたはリソースの操作が正常かどうかを定義するオブジェクトの、 2 つ以上の異なる状態の限定集合を表すメジャーである。これらの状態の 1 つが通常で、その他の状態はすべて通常ではないと見なされます。状態ベースのヘルス・インディケーターには、通常およびアテンションの 2 つの有効な状態があります。
- **コレクション状態ベースの**インディケーターは、データベース中の 1 つ以上のオブジェクトのコレクション状態を表すデータベース・レベルのメジャーである。コレクション中のオブジェクトごとにデータが取り込まれ、これらのオブジェクトの間で最も重大な条件が集約状態として表されます。コレクション中の 1 つ以

上のオブジェクトがアラートを必要とする状態である場合は、ヘルス・インディケーターはアテンションを表示します。コレクション状態ベースのヘルス・インディケーターには、通常およびアテンションの 2 つの有効な状態があります。

ヘルス・インディケーターは、インスタンス、データベース、表スペース、および表スペース・コンテナー・レベルです。

ヘルス・モニター情報には、ヘルス・センター、CLP、または API を介してアクセスできます。これらのツールを使用してヘルス・インディケーターの構成も行えます。

アラートは、正常の状態から正ではない状態への変化、または定義されたしきい値の境界に基づくヘルス・インディケーター値の警告またはアラームのゾーンへの変化に応答して生成されます。アラートには、アテンション、警告、およびアラームの 3 つがあります。

- 特定の状態を測定するヘルス・インディケーターの場合、通常でない状態が登録されると、アテンション・アラートが発行される。
- 値の連続範囲を計測するヘルス・インディケーターの場合は、正常、警告、およびアラームの各状態の境界やゾーンは、しきい値によって定義される。例えば、値がアラーム・ゾーンとして定義されている値のしきい値範囲に入ると、アラームのアラートが発行されて、問題に対して即時に対処が必要であることを示します。

ヘルス・モニターは、特定のヘルス・インディケーターの特定のアラート条件が最初に現れたときにのみ通知を送信し、アクションを実行します。ヘルス・インディケーターが特定のアラート条件のまま留まる場合、追加の通知は送信されず、追加のアクションも実行されません。ヘルス・インディケーターがアラート条件を変更するか、または通常の状態に戻って再びアラート条件に入る場合、新たに通知が送信され、アクションが実行されます。

以下の表は、様々なリフレッシュ・インターバルにおけるヘルス・インディケーターと、ヘルス・インディケーターの状態に対するヘルス・モニターの応答の例を示しています。この例では、デフォルトの警告しきい値として 80 %、アラームしきい値として 90 % を使用しています。

表 48. 様々なリフレッシュ・インターバルにおけるヘルス・インディケーターの状態

リフレッシュ・インターバル	ts.ts_util (表スペースの使用率) ヘルス・インディケーターの値	ts.ts_util ヘルス・インディケーターの状態	ヘルス・モニターの応答
1	80	警告	警告の通知が送信され、警告アラート条件のアクションが実行される
2	81	警告	通知は送信されず、アクションは実行されない
3	75	正常	通知は送信されず、アクションは実行されない
4	85	警告	警告の通知が送信され、警告アラート条件のアクションが実行される

表 48. 様々なリフレッシュ・インターバルにおけるヘルス・インディケーターの状態 (続き)

リフレッシュ・インターバル	ts.ts_util (表スペースの使用率) ヘルス・インディケーターの値	ts.ts_util ヘルス・インディケーターの状態	ヘルス・モニターの応答
5	90	アラーム	アラームの通知が送信され、アラーム条件のアクションが実行される

ヘルス・モニター・インターフェースの論理データ・グループへのマッピング

次の表に、サポートされているヘルス・スナップショット要求のタイプをすべてリストします。

表 49. ヘルス・モニター・インターフェースの論理データ・グループへのマッピング

API 要求タイプ	CLP コマンド	SQL 表関数	論理データ・グループ
SQLMA_DB2	get health snapshot for dbm	HEALTH_DBM_INFO	db2
		HEALTH_DBM_HI	health_indicator
	get health snapshot for dbm show detail	HEALTH_DBM_HI_HIS	health_indicator_history
SQLMA_DBASE	get health snapshot for database on *dbname*	HEALTH_DB_INFO	dbase
		HEALTH_DB_HI	health_indicator
	get health snapshot for database on *dbname* show detail	HEALTH_DB_HI_HIS	health_indicator_history
SQLMA_DBASE with SQLM_HMON_OPT_COLL_FULL in the agent_id	get health snapshot for database on *dbname* with full collection	HEALTH_DB_HIC	health_indicator, hi_obj_list
	get health snapshot for database on *dbname* show detail with full collection	HEALTH_DB_HIC_HIST	health_indicator_history, hi_obj_list
SQLMA_DBASE_ALL	get health snapshot for all databases	HEALTH_DB_INFO	dbase
		HEALTH_DB_HI	health_indicator
	get health snapshot for all databases show detail	HEALTH_DB_HI_HIS	health_indicator_history
SQLMA_DBASE_TABLESPACES	get health snapshot for tablespaces on *dbname*	HEALTH_TS_INFO	tablespace
		HEALTH_TS_HI	health_indicator
		HEALTH_CONT_INFO	tablespace_container
		HEALTH_CONT_HI	health_indicator
	get health snapshot for tablespaces on *dbname* show detail	HEALTH_TS_HI_HIS	health_indicator_history
		HEALTH_CONT_HI_HIS	health_indicator_history

以下の図は、論理データ・グループがヘルス・スナップショット・データ・ストリーム内に現れる順番を示しています。

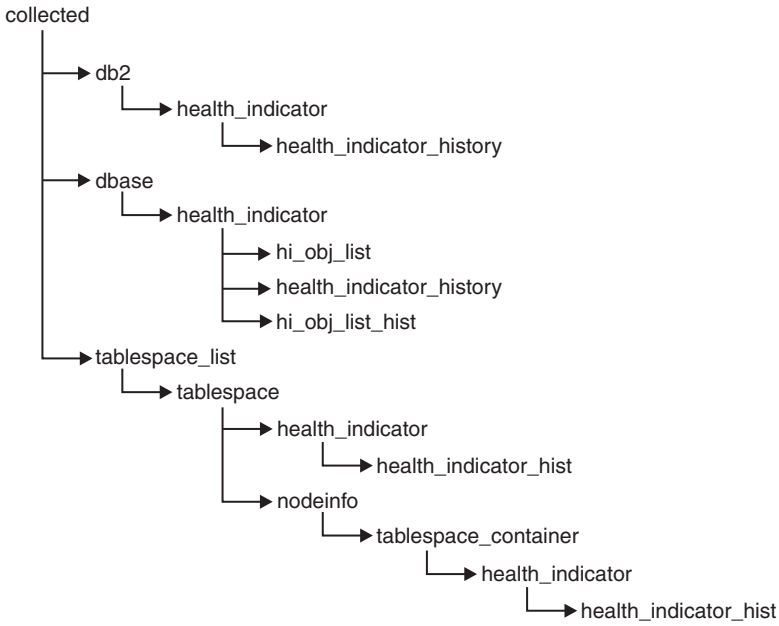


図 6. ヘルス・スナップショットの論理データ・グループ

ヘルス・インディケーターの要約

次の表には、すべてのヘルス・インディケーターが、カテゴリー別にグループ化されてリストされています。

表 50. データベース自動ストレージ使用率のヘルス・インディケーター

名前	ID	追加情報
データベース自動ストレージ使用率	db.auto_storage_util	188 ページの『db.auto_storage_util データベース自動ストレージ使用率 : ヘルス・インディケーター』

表 51. 表スペース・ストレージのヘルス・インディケーター

名前	ID	追加情報
表スペース自動サイズ変更状態	ts.ts_auto_resize_status	189 ページの『ts.ts_auto_resize_status 表スペース自動サイズ変更状態 : ヘルス・インディケーター』
自動サイズ変更表スペース使用率	ts.ts_util_auto_resize	190 ページの『ts.ts_util_auto_resize 自動サイズ変更表スペース使用率 : ヘルス・インディケーター』
表スペース使用率	ts.ts_util	190 ページの『ts.ts_util 表スペース使用率 :』
表スペース・コンテナー使用率	tsc.tscont_util	191 ページの『tsc.tscont_util 表スペース・コンテナー使用率 :』
表スペース操作可能状態	ts.ts_op_status	192 ページの『ts.ts_op_status 表スペース操作可能状態 :』

表 51. 表スペース・ストレージのヘルス・インディケーター (続き)

名前	ID	追加情報
表スペース・コンテナー操作可能状態	tsc.tscont_op_status	192 ページの『tsc.tscont_op_status 表スペース・コンテナー操作可能状態 :』
表スペース自動サイズ変更状態	ts.ts_auto_resize_status	189 ページの『ts.ts_auto_resize_status 表スペース自動サイズ変更状態 : ヘルス・インディケーター』

表 52. ソートのヘルス・インディケーター

名前	ID	追加情報
専用ソート・メモリー使用率	db2.sort_privmem_util	192 ページの『db2.sort_privmem_util 専用ソート・メモリー使用率 :』
共有ソート・メモリー使用率	db.sort_shrmem_util	193 ページの『db.sort_shrmem_util 共有ソート・メモリー使用率 :』
オーバーフローしたソートのパーセンテージ	db.spilled_sorts	194 ページの『db.spilled_sorts オーバーフローしたソートのパーセンテージ :』
長期共有ソート・メモリー使用率	db.max_sort_shrmem_util	194 ページの『db.max_sort_shrmem_util 長期共有ソート・メモリー使用率』

表 53. データベース・マネージャーのヘルス・インディケーター

名前	ID	追加情報
インスタンス操作可能状態	db2.db2_op_status	195 ページの『db2.db2_op_status インスタンス操作可能状態 :』
インスタンス最大重大度アラート状態	–	195 ページの『インスタンス最大重大度アラート状態 :』

表 54. データベースのヘルス・インディケーター

名前	ID	追加情報
データベース操作可能状態	db.db_op_status	196 ページの『db.db_op_status データベース操作可能状態 :』
データベース最大重大度アラート状態	–	196 ページの『データベース最大重大度アラート状態 :』

表 55. 保守のヘルス・インディケーター

名前	ID	追加情報
再編成の必要性	db.tb_reorg_req	197 ページの『db.tb_reorg_req 再編成の必要性 :』
統計収集の必要性ヘルス・インディケーター	db.tb_runstats_req	197 ページの『db.tb_runstats_req 統計収集の必要性 :』
データベース・バックアップの必要性	db.db_backup_req	198 ページの『db.db_backup_req データベース・バックアップの必要性 :』

表 56. 高可用性災害時リカバリーのヘルス・インディケーター

名前	ID	追加情報
HADR 操作可能状態ヘルス・インディケーター	db.hadr_op_status	198 ページの『db.hadr_op_status HADR 操作可能状態 :』
HADR ログ遅延ヘルス・インディケーター	db.hadr_delay	199 ページの『db.hadr_delay HADR ログ遅延 :』

表 57. ロギングのヘルス・インディケーター

名前	ID	追加情報
ログ使用率	db.log_util	199 ページの『db.log_util ログ使用率 :』
ログ・ファイル・システム使用率	db.log_fs_util	199 ページの『db.log_fs_util ログ・ファイル・システム使用率 :』

表 58. アプリケーション並行性のヘルス・インディケーター

名前	ID	追加情報
デッドロック率	db.deadlock_rate	200 ページの『db.deadlock_rate デッドロック率 :』
ロック・リスト使用率	db.locklist_util	201 ページの『db.locklist_util ロック・リスト使用率 :』
ロック・エスカレーション率	db.lock_escal_rate	202 ページの『db.lock_escal_rate ロック・エスカレーション率 :』
ロック待機中のアプリケーションのパーセンテージ	db.apps_waiting_locks	202 ページの『db.apps_waiting_locks ロック待機中のアプリケーションのパーセンテージ :』

表 59. パッケージ・キャッシュ、カタログ・キャッシュ、ワークスペースのヘルス・インディケーター

名前	ID	追加情報
カタログ・キャッシュ・ヒット率	db.catcache_hitratio	203 ページの『db.catcache_hitratio カタログ・キャッシュ・ヒット率 :』
パッケージ・キャッシュ・ヒット率	db.pkgcache_hitratio	203 ページの『db.pkgcache_hitratio パッケージ・キャッシュ・ヒット率 :』
共有ワークスペース・ヒット率	db.shrworkspace_hitratio	204 ページの『db.shrworkspace_hitratio 共有ワークスペース・ヒット率 :』

表 60. メモリーのヘルス・インディケーター

名前	ID	追加情報
モニター・ヒープ使用率	db2.mon_heap_util	204 ページの『db2.mon_heap_util モニター・ヒープ使用率 :』
データベース・ヒープ使用率	db.db_heap_util	204 ページの『db.db_heap_util データベース・ヒープ使用率 :』

表 61. フェデレーテッドのヘルス・インディケーター

名前	ID	追加情報
ニックネームの状態	db.fed_nicknames_op_status	205 ページの『db.fed_nicknames_op_status ニックネームの状態 :』
データ・ソース・サーバーの状態	db.fed_servers_op_status	205 ページの『db.fed_servers_op_status データ・ソース・サーバーの状態 :』

ヘルス・インディケーターの形式:

ヘルス・インディケーターによって収集されるデータの説明。

ヘルス・インディケーターのドキュメンテーションは次の標準形式で記述されます。

ID ヘルス・インディケーターの名前。この ID は、CLP からの構成で使用されます。

ヘルス・モニター・レベル
ヘルス・モニターによってヘルス・インディケーターがキャプチャーされる際のレベル。

分類 ヘルス・インディケーターのカテゴリー。

タイプ ヘルス・インディケーターのタイプ。以下の 4 つのタイプがあります。

- 上限しきい値ベース。アラートを生成する進行状況は正常、警告、アラームです。
- 下限しきい値ベース
- 状態ベース。 1 つの状態が正常で、その他の状態はすべて正常ではありません。
- コレクション状態ベース。状態は、集合中のオブジェクトの状態の集約に基づいています。

単位 パーセンテージなどの、ヘルス・インディケーターで測定されるデータの単位。状態ベースやコレクション状態ベースのヘルス・インディケーターには該当しません。

表スペース・ストレージのヘルス・インディケーター:

DMS 表スペースのヘルス・インディケーター:
この表は、表スペースの特性に基づき、どの表スペース・ヘルス・インディケーターが DMS 表スペースに関係しているかを説明します。

表 62. DMS 表スペースに関係する表スペース・ヘルス・インディケーター

表スペースの特性	定義されている表スペースの最大サイズ	定義されていない表スペースの最大サイズ
自動サイズ変更が使用可能 = Yes	ts.ts_util_auto_resize - 使用されている表スペースのパーセントを、ユーザーが定義した最大サイズと比べて追跡します。アラートは、表スペースがまもなくいっぱいになり、ユーザーによる介入が必要であることを示します。最大サイズが妥当な値に設定されている限り (つまり、最大サイズで指定されたスペース量が存在しているなら)、この構成に最も重要なヘルス・インディケーターです。 ts.ts_util - 現在割り振られている表スペース・ストレージの使用量を追跡します。表スペースがいっぱいになると表スペースはサイズを大きくしようとするため、アラートが出されても、問題を解決するためにユーザーによる介入を必要としないことがあります。 ts.ts_auto_resize_status - サイズ変更の試行が正常かどうかを追跡します。アラートは、表スペースがサイズ変更に失敗した (つまり、表スペースがいっぱいである) ことを示します。	ts.ts_util_auto_resize - 適用されません。表スペース・サイズの上限が指定されていません。 ts.ts_util - 現在割り振られている表スペース・ストレージの使用量を追跡します。表スペースはサイズを大きくしようとするため、アラートが出されても、問題を解決するためにユーザーによる介入を必要としないことがあります。 ts.ts_auto_resize_status - サイズ変更の試行が正常かどうかを追跡します。アラートは、表スペースがサイズ変更に失敗した (つまり、表スペースがいっぱいである) ことを示します。 **注:** DMS 表スペースが自動ストレージを使用して定義されており、最大サイズが指定されていない場合、db.auto_storage_util ヘルス・インディケーターにも注目する必要があります。このヘルス・インディケーターは、データベース・ストレージ・パスに関連したスペースの使用率を追跡します。このスペースがいっぱいになると、表スペースを大きくすることができません。その結果、表スペースのフル状態になることがあります。
自動サイズ変更が使用可能 = No	有効な構成ではありません。表スペースの最大サイズは、自動サイズ変更が使用可能になっている表スペースにのみ有効です。	ts.ts_util_auto_resize - 適用されません。表スペースはサイズ変更を試行しません。 ts.ts_util - 現在割り振られている表スペース・ストレージの使用量を追跡します。アラートは表スペースのフル状態を示しており、ユーザーによる即時介入が必要です。表スペースは自動的にサイズ変更を試行しません。 ts.ts_auto_resize_status - 適用されません。表スペースはサイズ変更を試行しません。

***db.auto_storage_util* データベース自動ストレージ使用率 : ヘルス・インディケーター:**

このヘルス・インディケーターは、定義されたデータベース・ストレージ・パスのストレージの使用量を追跡します。

ID db.auto_storage_util

ヘルス・モニター・レベル
データベース

分類 データベース

タイプ 上限しきい値ベース

単位 パーセンテージ

自動ストレージ表スペースが作成されると、データベース・ストレージ・パス上にそれらの表スペース用のコンテナーが自動的に割り振られます。データベース・ストレージ・パスが定義されているどのファイル・システム上にもスペースが残されていない場合、自動ストレージ表スペースはサイズを大きくすることができず、いっぱいになります。

インディケーターは、次の公式を使用して計算されます。

```
(db.auto_storage_used / db.auto_storage_total) * 100
```

ここで、

- *db.auto_storage_used* は、データベース・ストレージ・パスのリストで指定されているすべての物理ファイル・システムの使用中のスペースの合計です。
- *db.auto_storage_total* は、データベース・ストレージ・パスのリストで指定されているすべての物理ファイル・システムの全スペースの合計です。

データベース自動ストレージ・パス使用率は、データベース・ストレージ・パスのファイル・システム上で消費されたスペースのパーセントとして測定され、高いパーセントはこのインディケーターの最適関数を下回っていることを示します。

追加情報のフルになるまでの残り時間の計算は、すべてのフリー・スペースが消費されるまでの残りの時間を予測したものです。

ts.ts_auto_resize_status 表スペース自動サイズ変更状態 : ヘルス・インディケーター :

このヘルス・インディケーターは、自動サイズ変更が使用可能になっている DMS 表スペースに対して表スペースのサイズ変更操作が正常に行われているかどうかを識別します。自動サイズ変更が使用可能になっている DMS 表スペースがサイズを大きくできない場合、その表スペースは事実上いっぱいになっています。この状態は、表スペース・コンテナーが定義されているファイル・システム上にフリー・スペースがないか、または表スペース・コンテナーの自動サイズ変更設定の結果として起こります。例えば、定義された最大サイズに達している可能性や、増加量の設定が大きすぎるために残りのフリー・スペースでは収まらない可能性があります。

ID ts.ts_auto_resize_status

ヘルス・モニター・レベル
表スペース

分類 表スペース・ストレージ

タイプ 状態ベース

単位 適用外

ts.ts_util_auto_resize 自動サイズ変更表スペース使用率 : ヘルス・インディケーター :

このヘルス・インディケーターは、最大サイズが定義されている自動サイズ変更 DMS 表スペースごとのストレージの使用量を追跡します。最大サイズに達している場合、DMS 表スペースがいっぱいであると見なされます。

ID ts.ts_util_auto_resize

ヘルス・モニター・レベル
表スペース

分類 表スペース・ストレージ

タイプ 上限しきい値ベース

単位 パーセンテージ

インディケーターは、次の公式を使用して計算されます。

```
((ts.used * ts.page_size) / ts.max_size) * 100
```

ここで、

- *ts.used* は、 788 ページの『tablespace_used_pages 表スペース内の使用されているページ数 : モニター・エレメント』 の値です
- *ts.page_size* は、 778 ページの『tablespace_page_size - 表スペースのページ・サイズ : モニター・エレメント』 の値です
- *ts.max_size* は、 775 ページの『tablespace_max_size 表スペースの最大サイズ』 の値です

自動サイズ変更 DMS 表スペース使用率は、消費された最大表スペース・ストレージのパーセントとして測定されます。高いパーセントは、表スペースがいっぱいになろうとしていることを示します。この標識の追加情報に含まれる短期間と長期間の増加率は、現行増加率が短期間の例外的なものであるか、長期間にわたる一貫した増加であるかを判別するのに使用されます。

追加情報のフルになるまでの残り時間の計算は、最大サイズに達するまでの残りの時間を予測したものです。

ts.ts_util 表スペース使用率 ::

このヘルス・インディケーターは、各 DMS 表スペースのストレージの使用量を追跡します。

ID ts.ts_util

ヘルス・モニター・レベル
表スペース

分類 表スペース・ストレージ

タイプ 上限しきい値ベース

単位 パーセンテージ

すべてのコンテナーがフルの場合は、DMS 表スペースはフルであると考えられます。

表スペースで自動サイズ変更が使用可能になっている場合、このヘルス・インディケーターは評価されません。代わりに、データベース自動ストレージ使用率 **db.auto_storage_util** と表スペース自動サイズ変更状態 (**ts.ts_auto_resize_status**) ヘルス・インディケーターが表スペースのストレージのモニターに関係します。この表スペースで最大サイズを定義した場合には、自動サイズ変更表スペース使用率 (**ts.ts_util_auto_resize**) ヘルス・インディケーターも使用可能になります。表スペース使用率のパーセンテージは、TBSP_UTILIZATION 管理ビューの TBSP_UTILIZATION_PERCENT 列から取得できます (必要な場合)。

インディケーターは、次の公式を使用して計算されます。

```
(ts.used / ts.usable) * 100
```

ここで、

- *ts.used* は、 788 ページの『tablespace_used_pages 表スペース内の使用されているページ数 : モニター・エレメント』 の値です
- *ts.usable* は、 787 ページの『tablespace_usable_pages 表スペース内の使用可能ページ数 : モニター・エレメント』 の値です

表スペース使用率は消費されたスペースのパーセントとして測定され、高いパーセントはこの標識の最適関数を下回っていることを示します。

この標識の追加情報に含まれる短期間と長期間の増加率は、現行増加率が短期間の例外的なものであるか、長期間にわたる一貫した増加であるかを判別するのに使用されます。

追加情報のフルになるまでの残り時間の計算は、すべてのフリー・スペースが消費されるまでの残りの時間を予測したものです。

tsc.tscont_util 表スペース・コンテナー使用率 ::

このヘルス・インディケーターは、自動ストレージを使用していない各 SMS 表スペースのストレージの使用量を追跡します。

ID tsc.tscont_util

ヘルス・モニター・レベル
表スペース・コンテナー

分類 表スペース・ストレージ

タイプ 上限しきい値ベース

単位 パーセンテージ

コンテナーが定義されているファイル・システムにスペースが残されていない場合は、 SMS 表スペースはフルであると考えられます。

SMS コンテナーを拡張するためのフリー・スペースがファイル・システムで使用不可である場合は、関連する表スペースがフルになります。

フリー・スペースを消費するファイル・システムに定義されている各コンテナーに対してアラートが発行される場合があります。

インディケーターは、次の公式を使用して計算されます。

```
(fs.used / fs.total)*100
```

ここで fs はコンテナーがあるファイル・システムです。

SMS 表スペース使用率は消費されたスペースのパーセントとして測定され、高いパーセントはこの標識の最適関数を下回っていることを示します。

この標識の追加情報に含まれる短期間と長期間の増加率は、現行増加率が短期間の例外的なものであるか、長期間にわたる一貫した増加であるかを判別するのに使用されます。

追加情報のフルになるまでの残り時間の計算は、すべてのフリー・スペースが消費されるまでの残りの時間を予測したものです。

ts.ts_op_status 表スペース操作可能状態 ::

表スペースの状態は、実行できるアクティビティーまたはタスクを制約します。正常から他の状態に変わると、アテンション・アラートが生成される場合があります。

ID ts.ts_op_status

ヘルス・モニター・レベル
表スペース

分類 表スペース・ストレージ

タイプ 状態ベース

単位 適用外

tsc.tscont_op_status 表スペース・コンテナー操作可能状態 ::

このヘルス・インディケーターは、表スペース・コンテナーのアクセス可能性を追跡します。コンテナーのアクセス可能性は、実行できるアクティビティーまたはタスクを制約します。コンテナーがアクセス不能の場合は、アテンション・アラートが生成される場合があります。

ID tsc.tscont_op_status

ヘルス・モニター・レベル
表スペース・コンテナー

分類 表スペース・ストレージ

タイプ 状態ベース

単位 適用外

ソートのヘルス・インディケーター:

db2.sort_privmem_util 専用ソート・メモリー使用率 ::

この標識は、専用ソート・メモリーの使用率を追跡します。 db2.sort_heap_allocated (システム・モニター・エレメント) >= *sheapthres* (DBM 構成パラメーター) の場合は、ソートは *sortheap* パラメーターで定義されているソート・ヒープを十分に取得していない可能性があり、アラートが生成される場合があります。

ID db2.sort_privmem_util

ヘルス・モニター・レベル
データベース

分類 ソート

タイプ 上限しきい値ベース

単位 パーセンテージ

ソートを行うためのヒープ・スペースが十分あり、ソートが不必要にオーバーフローしていなければ、ソートは正常稼働していると考えられます。

インディケーターは、次の公式を使用して計算されます。

```
(db2.sort_heap_allocated / sheapthres)*100
```

「ポストしきい値ソート」スナップショット・モニター・エレメントは、ソート・ヒープしきい値を超えた後に要求されたソート数を測定します。追加の詳細に表示されるこのインディケーターの値は、このヘルス・インディケーターの問題の重大度を示します。

「使用された最大専用ソート・メモリー」スナップショット・モニター・エレメントは、インスタンスの専用ソート・メモリー最高水準点を保持します。追加情報に示されるこの標識の値は、インスタンスが最後に再生された後の任意の時点で使用中だった専用ソート・メモリーの最大量を示します。この値は *sheapthres* の適切な値を決定するために利用できます。

db.sort_shrmem_util 共有ソート・メモリー使用率 ::

この標識は、共有ソート・メモリーの使用率を追跡します。 *sheapthres_shr* データベース構成パラメーターはハード・リミットです。割り振りが制限に近くなると、アラートが生成される場合があります。

ID db.sort_shrmem_util

ヘルス・モニター・レベル
データベース

分類 ソート

タイプ 上限しきい値ベース

単位 パーセンテージ

ソートを行うためのヒープ・スペースが十分あり、ソートが不必要にオーバーフローしていなければ、ソートは正常稼働していると考えられます。

インディケーターは、次の公式を使用して計算されます。

```
(db.sort_shrheap_allocated / sheapthres_shr)*100
```

sheapthres_shr が 0 に設定されると、 *sheapthres* は共有ソート・ヒープしきい値として使用されることに注意してください。

「使用された最大共有ソート・メモリー」スナップショット・モニター・エレメントは、データベースの共有ソート・メモリー最高水準点を保持します。追加情報に表示されるこの標識の値は、データベースがアクティブになった後の任意の時点で

使用中であった共有ソート・メモリーの最大量を示します。この値は共有ソート・メモリーしきい値の適切な値を決定するために利用できます。

現在のワークロードの必要に応じてソート・メモリー・リソースが自動的に割り振られるよう、メモリーのセルフチューニング・フィーチャーの使用を考慮してください。ソート・メモリー領域でメモリーのセルフチューニング・フィーチャーが使用可能である場合、このヘルス・インディケーターがしきい値検査を使用不可にするように構成する必要があります。

***db.spilled_sorts* オーバーフローしたソートのパーセンテージ ::**

ディスクにオーバーフローするソートは、重大なパフォーマンス低下の原因となる可能性があります。これが起こると、アラートが生成される場合があります。

ID db.spilled_sorts

ヘルス・モニター・レベル
データベース

分類 ソート

タイプ 上限しきい値ベース

単位 パーセンテージ

ソートを行うためのヒープ・スペースが十分あり、ソートが不必要にオーバーフローしていなければ、ソートは正常稼働していると考えられます。

インディケーターは、次の公式を使用して計算されます。

$$(db.sort\_overflows_{t} - db.sort\_overflows_{t-1}) / (db.total\_sorts_{t} - db.total\_sorts_{t-1}) * 100$$

t は現在のスナップショットで、*t-1* は 1 時間前のスナップショットです。システム・モニター・エレメント db.sort_overflows (sort_overflows モニター・エレメントが基になる) はソート・ヒープを使い果たし、一時記憶域のディスク・スペースを必要とした可能性のあるソートの合計数です。エレメント db.total_sorts (total_sorts モニター・エレメントが基になる) は実行されたソートの合計数です。

現在のワークロードの必要に応じてソート・メモリー・リソースが自動的に割り振られるよう、メモリーのセルフチューニング・フィーチャーの使用を考慮してください。ソート・メモリー領域でメモリーのセルフチューニング・フィーチャーが使用可能である場合、このヘルス・インディケーターがしきい値検査を使用不可にするように構成する必要があります。

***db.max_sort_shrmem_util* 長期共有ソート・メモリー使用率:**

この標識は構成済み共有ソート・ヒープを追跡し、 DB2 データベース・システムの他のどこかでの使用のために解放できるリソースがないかどうかを調べます。

ID db.max_sort_shrmem_util

ヘルス・モニター・レベル
データベース

分類 ソート

タイプ 下限しきい値ベース

単位　パーセンテージ

ソートを行うためのヒープ・スペースが十分あり、ソートが不必要にオーバーフローしていなければ、ソートは正常稼働していると考えられます。

使用量のパーセンテージが低い場合はアラートが生成される場合があります。

インディケーターは、次の公式を使用して計算されます。

```
(db.max_shr_sort_mem / sheapthres_shr)*100
```

システム・モニター・エレメント db.max_shr_sort_mem (sort_shrheap_top モニター・エレメントが基になる) は、共有ソート・メモリー使用量の最高水準点です。

現在のワークロードの必要に応じてソート・メモリー・リソースが自動的に割り振られるよう、メモリーのセルフチューニング・フィーチャーの使用を考慮してください。ソート・メモリー領域でメモリーのセルフチューニング・フィーチャーが使用可能である場合、このヘルス・インディケーターがしきい値検査を使用不可にするように構成する必要があります。

データベース・マネージャー (DBMS) のヘルス・インディケーター:

db2.db2_op_status インスタンス操作可能状態 ::

インスタンス状態が実行されているアクティビティーまたはタスクを制約していないときは、インスタンスは正常稼働していると考えられます。

ID　db2.db2_op_status

ヘルス・モニター・レベル
インスタンス

分類　DBMS

タイプ　状態ベース

単位　適用外

状態は、アクティブ、静止ペンディング、静止、またはダウンのいずれかになります。非アクティブの状態では、アテンション・アラートが生成される場合があります。

db2.db2_op_status ヘルス・インディケーターがダウン状態に入ると、ヘルス・モニターはこのインディケーターに対するアクションを実行できません。この状態の原因になりうるのは、例えば、明示的な停止要求または異常終了に起因して、インディケーターがモニターしているインスタンスが非アクティブになった場合です。異常終了後、常にインスタンスを自動的に再始動させるには、インスタンスの操作可能状態を維持できるよう障害モニター (db2fm) を構成することができます。

インスタンス最大重大度アラート状態 ::

この標識は、モニター対象のインスタンスのロールアップ・アラート状態を表します。インスタンスのアラート状態は、インスタンスとそのデータベース、およびモニター対象のデータベース・オブジェクトの最高レベルのアラート状態です。

ID 該当なし。このヘルス・インディケーターには、構成または推奨サポートはありません。

ヘルス・モニター・レベル
インスタンス

分類 DBMS

タイプ 状態ベース

単位 適用外

アラート状態の順序は次のようになります。

- アラーム
- 警告
- アテンション
- 正常

インスタンスのアラート状態によって、DB2 データベース・システムが全体として正常稼働しているかどうかが決まります。

データベースのヘルス・インディケーター:

db.db_op_status データベース操作可能状態 ::

データベースの状態は、実行できるアクティビティーまたはタスクを制約します。状態は、アクティブ、静止ペンディング、静止、またはロールフォワードのいずれかになります。アクティブから他の状態に変わると、アテンション・アラートが生成される場合があります。

ID db.db_op_status

ヘルス・モニター・レベル
データベース

分類 データベース

タイプ 状態ベース

単位 適用外

データベース最大重大度アラート状態 ::

この標識は、モニター対象のデータベースのロールアップ・アラート状態を表します。データベースのアラート状態は、データベースとそのオブジェクトの最高レベルのアラート状態です。

ID 該当なし。このヘルス・インディケーターには、構成または推奨サポートはありません。

ヘルス・モニター・レベル
データベース

分類 データベース

タイプ 状態ベース

単位 適用外

アラート状態の順序は次のようになります。

- アラーム
- 警告
- アテンション
- 正常

保守のヘルス・インディケーター:

db.tb_reorg_req 再編成の必要性 ::

このヘルス・インディケーターは、データベース中の表や索引を再編成する必要性を追跡します。フラグメント化されたデータを除去するには、表か、表に定義されたすべての索引を再編成する必要があります。再編成するには、情報を圧縮して、行か索引データを再構成します。その結果、パフォーマンスが向上し、表や索引中のフリー・スペースが増えます。

ID db.tb_reorg_req

ヘルス・モニター・レベル
データベース

分類 データベース保守

タイプ コレクション状態ベース

単位 適用外

ご使用の自動保守ポリシーに、評価対象となる表の名前を指定することによって、このヘルス・インディケーターにより評価される表集合をフィルターに掛けることができます。これは、「自動保守」ウィザードを使用して行えます。

アテンション・アラートが生成されて、再編成が必要なことが示される場合もあります。 AUTO_REORG データベース構成パラメーターを ON に設定すると、再編成を自動化できます。自動再編成を使用可能にした場合、アテンション・アラートにより、 1 つ以上の自動再編成が正常に完了できなかったこと、あるいは、再編成を必要とする表があるものの、データベース・パーティションごとの表のサイズが、オフライン再編成を考慮すべき表の最大サイズ基準を超えているために、自動再編成が実行されていないことが示されます。注意が必要なオブジェクトのリストについては、このヘルス・インディケーターの集合の詳細を参照してください。

db.tb_runstats_req 統計収集の必要性 ::

このヘルス・インディケーターは、データベース中の表やそれらの索引の統計を収集する必要性を追跡します。照会の実行時間を改善するには、表および表に定義されたすべての索引に統計が必要です。

ID db.tb_runstats_req

ヘルス・モニター・レベル
データベース

分類 データベース保守

タイプ コレクション状態ベース

単位 適用外

SQL 照会を使用して、このヘルス・インディケーターによって考慮する表を限定できます。この照会のシステム表に対する副選択節が、追加情報中の有効範囲に示されます。

アテンション・アラートが生成されて、統計の収集を促される場合もあります。AUTO_RUNSTATS データベース構成パラメーターを ON に設定すると、統計を自動的に収集できます。統計の自動収集を使用可能にすると、アテンション・アラートにより、 1 つ以上の統計の自動収集アクションが正常に完了しなかったことが示されます。

db.db_backup_req データベース・バックアップの必要性 ::

このヘルス・インディケーターは、データベースのバックアップの必要性を追跡します。ハードウェアやソフトウェアの障害の場合にデータが消失する可能性から保護するために、リカバリー計画の一部として、定期的にバックアップを取る必要があります。

ID db.db_backup_req

ヘルス・モニター・レベル
データベース

分類 データベース保守

タイプ 状態ベース

単位 適用外

このヘルス・インディケーターは、経過時間と、前回のバックアップ以後に変更されたデータの量に基づいて、データベースのバックアップが必要な時点を判別します。

アテンション・アラートが生成されて、データベースのバックアップが必要なことが示される場合もあります。 AUTO_DB_BACKUP データベース構成パラメーターを ON に設定すると、データベースのバックアップを自動化できます。自動データベース・バックアップを使用可能にすると、アテンション・アラートにより、 1 つ以上の自動データベース・バックアップが正常に完了しなかったことが示されます。

高可用性災害時リカバリー (HADR) ヘルス・インディケーター:

db.hadr_op_status HADR 操作可能状態 ::

このヘルス・インディケーターは、データベースの高可用性災害時リカバリー (HADR) 操作可能状態を追跡します。 1 次サーバーとスタンバイ・サーバーの間の状態は、接続済み、混雑、または切断のいずれかになります。接続済みから他の状態に変わると、アテンション・アラートが生成される場合があります。

ID db.hadr_op_status

ヘルス・モニター・レベル
データベース

分類 高可用性災害時リカバリー

タイプ 状態ベース

単位　適用外

db.hadr_delay HADR ログ遅延 ::

このヘルス・インディケーターは、1 次データベースに対するデータ変更と、スタンバイ・データベースに対するこれらの変更内容のレプリケーションの間の、現在の平均遅延 (分単位) を追跡します。遅延値が大きい場合は、 1 次データベースに障害が起きた後にスタンバイ・データベースにフェイルオーバーする際に、データ損失が生じる可能性があります。さらに遅延値が大きいと、 1 次データベースがスタンバイ・データベースより優先になっているためテークオーバーが必要な場合に、ダウン時間が長くなる可能性もあります。

ID　db.hadr_delay

ヘルス・モニター・レベル
データベース

分類　高可用性災害時リカバリー

タイプ　上限しきい値ベース

単位　分

ロギングのヘルス・インディケーター:

db.log_util ログ使用率 ::

このインディケーターは、データベースで使用されたアクティブ・ログ・スペースの合計量 (バイト数) を追跡します。

ID　db.log_util

ヘルス・モニター・レベル
データベース

分類　ロギング

タイプ　上限しきい値ベース

単位　パーセンテージ

ログ使用率は消費されたスペースのパーセントとして測定され、パーセンテージが高い場合はアラートが生成される場合があります。

インディケーターは、次の公式を使用して計算されます。

```
(db.total_log_used / (db.total_log_used + db.total_log_available)) * 100
```

追加情報に示されるログ関連のデータベース構成パラメーターの値は、ログの現行割り振りを表示します。追加情報には、最も古いアクティブ・トランザクションを持つアプリケーションのアプリケーション ID も含まれます。このアプリケーションにログ・スペースの解放を強制することができます。

db.log_fs_util ログ・ファイル・システム使用率 ::

ログ・ファイル・システム使用率は、トランザクション・ログが常駐するファイル・システムの使用率を追跡します。

ID　db.log_fs_util

ヘルス・モニター・レベル
データベース

分類 ロギング

タイプ 上限しきい値ベース

単位 パーセンテージ

ファイル・システムに空きがなければ、DB2 データベース・システムは新規ログ・ファイルを作成できない可能性があります。

ログ使用率は消費されたスペースのパーセントとして測定されます。ファイル・システムのフリー・スペースの量が最小の場合 (つまり使用率のパーセンテージが高い場合) は、アラートが生成される場合があります。

インディケーターは、次の公式を使用して計算されます: *(fs.log_fs_used / fs.log_fs_total)*100*。ここで fs はログが常駐するファイル・システムです。

追加情報に示されるログ関連のデータベース構成パラメーターの値は、ログの現行割り振りを表示します。ユーザー出口が使用可能であるかどうかも追加の詳細情報に示されます。

追加の詳細情報に示される「ディスクがフルになった時はログをブロック (Block on Log Disk Full)」が「はい (yes)」に設定され、使用率が 100% である場合、ログ・ファイルが正常に作成されるまでトランザクションをコミットできないアプリケーションへの影響を制限するために、アラートをできるだけ早く解決する必要があります。

アプリケーション並行性のヘルス・インディケーター:

db.deadlock_rate デッドロック率 ::

デッドロック率は、データベースでデッドロックが起きる率と、アプリケーションで競合問題が発生する度合いを追跡します。

ID db.deadlock_rate

ヘルス・モニター・レベル
データベース

分類 アプリケーション並行性

タイプ 上限しきい値ベース

単位 時間当たりのデッドロック数

デッドロックは次の状態が原因で起こることがあります。

- データベースでロック・エスカレーションが発生している場合。
- システムが生成した行のロッキング数が十分なときに、アプリケーションが表を明示的にロッキングしている場合。
- アプリケーションがバインディングのときに不適切な分離レベルを使用している場合。
- カタログ表が反復可能読み取りのためにロックされている場合。

- 複数のアプリケーションが同じロックを異なる順序で獲得しているために、デッドロックになっている場合。

インディケーターは、次の公式を使用して計算されます。

(db.deadlocks$_t$ - db.deadlocks$_{t-1}$)

ここで *t* は現在のスナップショットで、*t-1* は現在のスナップショットの 60 分前に取られた最後のスナップショットです。

デッドロック率が高くなると、アラートを生成する可能性のある競合の度合いも大きくなります。

db.locklist_util ロック・リスト使用率 ::

このインディケーターは、使用されているロック・リスト・メモリーの量を追跡します。

ID db.locklist_util

ヘルス・モニター・レベル
データベース

カテゴリー
アプリケーション並行性

タイプ 上限しきい値ベース

単位 パーセンテージ

データベースごとにロック・リストが 1 つあり、データベースに現在接続しているすべてのアプリケーションが保持しているロックが入っています。ロック・リスト・メモリーには設定限界があります。一度その限界に達すると、次の状態が原因でパフォーマンスが低下します。

- ロック・エスカレーションにより行ロックから表ロックへの変換が行われ、その結果、データベースの共有オブジェクトにおける並行性が低下する。
- アプリケーションによる限定された表ロック待ちのため、アプリケーション間でさらに多くのデッドロックが起きる。その結果、トランザクションがロールバックされます。

ロック要求の最大数がデータベースの限界設定に達すると、アプリケーションにエラーが戻されます。

インディケーターは、次の公式を使用して計算されます。

(db.lock_list_in_use / (*locklist* * 4096)) * 100

使用率は消費されたメモリーのパーセントとして測定され、パーセンテージが高い場合は正常稼働ではない状態を示します。

現在のワークロードの必要に応じてロック・メモリー・リソースが自動的に割り振られるよう、メモリーのセルフチューニング・フィーチャーの使用を考慮してください。ロック・メモリー領域でメモリーのセルフチューニング・フィーチャーが使用可能である場合、このヘルス・インディケーターがしきい値検査を使用不可にするように構成する必要があります。

db.lock_escal_rate **ロック・エスカレーション率 ::**

このインディケーターは、ロックが行ロックから表ロックにエスカレートされた率を追跡します。このエスカレーションの結果、トランザクションの並行性が影響を受けます。

ID db.lock_escal_rate

ヘルス・モニター・レベル
データベース

カテゴリー
アプリケーション並行性

タイプ 上限しきい値ベース

単位 時間当たりのロック・エスカレーション数

アプリケーションが保留するロックの合計数がそのアプリケーションで使用可能なロック・リスト・スペースの最大量に達した場合、またはすべてのアプリケーションが使用するロック・リスト・スペースが合計ロック・リスト・スペースに近くなると、ロックはエスカレートされます。使用可能なロック・リスト・スペースの量は、 *maxlocks* および *locklist* データベース構成パラメーターによって決まります。

アプリケーションが許可されているロックの最大数に達し、エスカレートするロックがない場合は、アプリケーションは他のアプリケーションに割り振られたロック・リストのスペースを使用します。データベースごとにロック・リストが 1 つあり、データベースに現在接続しているすべてのアプリケーションが保持しているロックが入っています。ロック・リスト全体が満杯になるとエラーが起こります。

インディケーターは、次の公式を使用して計算されます。

$(db.lock\_escals_{t} - db.lock\_escals_{t-1})$

ここで 't' は現在のスナップショットで、't-1' は現在のスナップショットの 60 分前に取られた最後のスナップショットです。

デッドロック率が高くなると、アラートを生成する可能性のある競合の度合いも大きくなります。

現在のワークロードの必要に応じてロック・メモリー・リソースが自動的に割り振られるよう、メモリーのセルフチューニング・フィーチャーの使用を考慮してください。ロック・メモリー領域でメモリーのセルフチューニング・フィーチャーが使用可能である場合、このヘルス・インディケーターがしきい値検査を使用不可にするように構成する必要があります。

db.apps_waiting_locks **ロック待機中のアプリケーションのパーセンテージ ::**

このインディケーターは、現在実行中でロック待ちのすべてのアプリケーションのパーセンテージを測定します。

ID db.apps_waiting_locks

ヘルス・モニター・レベル
データベース

分類 アプリケーション並行性

タイプ 上限しきい値ベース

単位 パーセンテージ

パーセンテージが高い場合は、パフォーマンスに悪影響を及ぼす並行性の問題がアプリケーションで発生していることを示します。

インディケーターは、次の公式を使用して計算されます。

```
(db.locks_waiting / db.appls_cur_cons) *100)
```

パッケージ・キャッシュ、カタログ・キャッシュ、ワークスペースのヘルス・インディケーター:

***db.catcache_hitratio* カタログ・キャッシュ・ヒット率 ::**

ヒット率は、カタログ・キャッシュを使用できることによりディスク上のカタログに実際にアクセスしないで済んでいる程度を示すパーセンテージです。高い率は、実際のディスク入出力アクセスの回避に成功していることを示します。

ID db.catcache_hitratio

ヘルス・モニター・レベル
データベース

分類 パッケージおよびカタログ・キャッシュ、およびワークスペース

タイプ 下限しきい値ベース

単位 パーセンテージ

インディケーターは、次の公式を使用して計算されます。

```
(1-(db.cat_cache_inserts/db.cat_cache_lookups))*100
```

***db.pkgcache_hitratio* パッケージ・キャッシュ・ヒット率 ::**

ヒット率は、パッケージ・キャッシュを使用できることによりシステム・カタログから静的 SQL のためのパッケージとセクションを再ロードしないで済み、また動的 SQL ステートメントを再コンパイルしないで済んでいる程度を示すパーセンテージです。高い率は、これらのアクティビティーの回避に成功していることを示します。

ID db.pkgcache_hitratio

ヘルス・モニター・レベル
データベース

分類 パッケージおよびカタログ・キャッシュ、およびワークスペース

タイプ 下限しきい値ベース

単位 パーセンテージ

インディケーターは、次の公式を使用して計算されます。

```
(1-(db.pkg_cache_inserts/db.pkg_cache_lookups))*100
```

現在のワークロードの必要に応じてパッケージ・キャッシュ・メモリー・リソースが自動的に割り振られるよう、メモリーのセルフチューニング・フィーチャーの使用を考慮してください。パッケージ・キャッシュ・メモリー領域でメモリーのセル

フチューニング・フィーチャーが使用可能である場合、このヘルス・インディケーターがしきい値検査を使用不可にするように構成する必要があります。

db.shrworkspace_hitratio 共有ワークスペース・ヒット率 ::

ヒット率は、共有 SQL ワークスペースを使用できることにより、実行されようとしている SQL ステートメントのセクションの初期化が不要になっている程度を示すパーセンテージです。高い率は、このアクティビティーの回避に成功していることを示します。

ID db.shrworkspace_hitratio

ヘルス・モニター・レベル
データベース

カテゴリー
パッケージおよびカタログ・キャッシュ、およびワークスペース

タイプ 下限しきい値ベース

単位 パーセンテージ

インディケーターは、次の公式を使用して計算されます。

```
(1-(db.shr_workspace_section_inserts/db.shr_workspace_section_lookups))*100
```

メモリーのヘルス・インディケーター:

db2.mon_heap_util モニター・ヒープ使用率 ::

このインディケーターは、ID が SQLM_HEAP_MONITOR のメモリー・プールを基に、モニター・ヒープ・メモリーの使用量を追跡します。

ID db2.mon_heap_util

ヘルス・モニター・レベル
インスタンス

分類 メモリー

タイプ 上限しきい値ベース

単位 パーセンテージ

使用率は次の公式を使用して計算されます。

```
(db2.pool_cur_size / db2.pool_max_size) * 100
```

これはメモリー・プール ID が SQLM_HEAP_MONITOR の場合です。

一度このパーセンテージが最大の 100% に達すると、モニター操作が失敗する場合があります。

db.db_heap_util データベース・ヒープ使用率 ::

このインディケーターは、ID が SQLM_HEAP_DATABASE のメモリー・プールを基に、モニター・ヒープ・メモリーの使用量を追跡します。

ID db.db_heap_util

ヘルス・モニター・レベル
データベース

分類 メモリー

タイプ 上限しきい値ベース

単位 パーセンテージ

使用率は次の公式を使用して計算されます。

```
(db.pool_cur_size / db.pool_max_size) * 100
```

これはメモリー・プール ID が SQLM_HEAP_DATABASE の場合です。

一度このパーセンテージが最大の 100% に達すると、使用可能なヒープがないため、照会および操作が失敗する場合があります。

フェデレーテッドのヘルス・インディケーター:

db.fed_nicknames_op_status ニックネームの状態 ::

このヘルス・インディケーターは、フェデレーテッド・データベース中で定義されているニックネームをすべて検査し、無効なニックネームがあるかどうかを判別します。データ・ソース・オブジェクトがドロップされたり変更が加えられたりした場合や、ユーザー・マッピングが誤っている場合には、ニックネームが無効になることがあります。

ID db.fed_nicknames_op_status

ヘルス・モニター・レベル
データベース

分類 フェデレーテッド

タイプ コレクション状態ベース

単位 適用外

フェデレーテッド・データベース中で定義されているニックネームが無効な場合は、アテンション・アラートが生成されることがあります。注意が必要なオブジェクトのリストについては、このヘルス・インディケーターの集合の詳細を参照してください。

このヘルス・インディケーターがニックネームの状況を検査するには、FEDERATED データベース・マネージャー・パラメーターを YES に設定しなければなりません。

db.fed_servers_op_status データ・ソース・サーバーの状態 ::

このヘルス・インディケーターは、フェデレーテッド・データベース中で定義されているデータ・ソース・サーバーをすべて検査し、使用できないものがあるかどうかを判別します。データ・ソース・ソース・サーバーが停止したり、存在しなくなったり、構成が誤っていたりすると、データ・ソース・サーバーが使用できないことがあります。

ID db.fed_servers_op_status

ヘルス・モニター・レベル
データベース

分類 フェデレーテッド

タイプ コレクション状態ベース

単位 適用外

フェデレーテッド・データベース中で定義されているニックネームが無効な場合は、アテンション・アラートが生成されることがあります。注意が必要なオブジェクトのリストについては、このヘルス・インディケーターの集合の詳細を参照してください。

このヘルス・インディケーターがデータ・ソース・サーバーの状況を検査するには、 FEDERATED データベース・マネージャー・パラメーターを YES に設定しなければなりません。

ヘルス・モニター・インターフェース

次の表は、API のヘルス・モニター・インターフェースをリストしています。

注: これらの API は推奨されておらず、ヘルス・モニターがバージョン 9.7 で推奨されなくなったため、今後のリリースでは除去される可能性があります。

表 63. ヘルス・モニター・インターフェース: API

モニター・タスク	API
ヘルス・スナップショットのキャプチャー	db2GetSnapshot - スナップショット・クラス SQLM_CLASS_HEALTH を指定してスナップショットを取得
集合オブジェクトの全リストを含むヘルス・スナップショットのキャプチャー	db2GetSnapshot - スナップショット・クラス SQLM_CLASS_HEALTH および agent_id に SQLM_HMON_OPT_COLL_FULL を指定してスナップショットを取得
公式、追加情報、および履歴を含むヘルス・スナップショットのキャプチャー	db2GetSnapshot - スナップショット・クラス SQLM_CLASS_HEALTH_WITH_DETAIL を指定してスナップショットを取得
公式、追加情報、履歴、および集合オブジェクトの全リストを含むヘルス・スナップショットのキャプチャー	db2GetSnapshot - スナップショット・クラス SQLM_CLASS_HEALTH_WITH_DETAIL および agent_id に SQLM_HMON_OPT_COLL_FULL を指定してスナップショットを取得
自己記述型データ・ストリームの変換	db2ConvMonStream - モニター・ストリームの変換
ヘルス・スナップショットのサイズ見積もり	db2GetSnapshotSize - db2GetSnapshot 出力バッファーに必要なサイズの見積もり

次の表は、CLP コマンドのヘルス・モニター・インターフェースをリストしています。

注: これらのコマンドは推奨されておらず、ヘルス・モニターがバージョン 9.7 で推奨されなくなったため、今後のリリースでは除去される可能性があります。

表 64. ヘルス・モニター・インターフェース: CLP コマンド

モニター・タスク	CLP コマンド
ヘルス・スナップショットのキャプチャー	GET HEALTH SNAPSHOT コマンド
公式、追加情報、および履歴を含むヘルス・スナップショットのキャプチャー	GET HEALTH SNAPSHOT WITH DETAILS コマンド

次の表は、SQL 関数のヘルス・モニター・インターフェースをリストしています。

注: これらの SQL 関数は推奨されておらず、ヘルス・モニターがバージョン 9.7 で推奨されなくなったため、今後のリリースでは除去される可能性があります。

表 65. ヘルス・モニター・インターフェース: SQL 関数

モニター・タスク	SQL 関数
データベース・マネージャー・レベルの正常性に関する情報のスナップショット	HEALTH_DBM_INFO
データベース・マネージャー・レベルのヘルス・インディケーターのスナップショット	HEALTH_DBM_HI
データベース・マネージャー・レベルのヘルス・インディケーター履歴のスナップショット	HEALTH_DBM_HI_HIS
データベース・レベルの正常性に関する情報のスナップショット	HEALTH_DB_INFO
データベース・レベルのヘルス・インディケーターのスナップショット	HEALTH_DB_HI
データベース・レベルのヘルス・インディケーター履歴のスナップショット	HEALTH_DB_HI_HIS
データベース・レベルのヘルス・インディケーター集合のスナップショット	HEALTH_DB_HIC
データベース・レベルのヘルス・インディケーター収集履歴のスナップショット	HEALTH_DB_HIC_HIS
表スペース・レベルの正常性に関する情報のスナップショット	HEALTH_TBS_INFO
表スペース・レベルのヘルス・インディケーターのスナップショット	HEALTH_TBS_HI
表スペース・レベルのヘルス・インディケーター履歴のスナップショット	HEALTH_TBS_HI_HIS
表スペース・コンテナー・レベルの正常性に関する情報のスナップショット	HEALTH_CONT_INFO
表スペース・コンテナー・レベルのヘルス・インディケーターのスナップショット	HEALTH_CONT_HI
表スペース・コンテナー・レベルのヘルス・インディケーター履歴のスナップショット	HEALTH_CONT_HI_HIS

ヘルス・モニター SQL 表関数:

次の表に、スナップショットの表関数をすべてリストします。それぞれの表関数は、ヘルス・スナップショット要求タイプに対応しています。

注: これらの表関数は推奨されておらず、ヘルス・モニターがバージョン 9.7 で推奨されなくなったため、今後のリリースでは除去される可能性があります。

表 66. スナップショット・モニター SQL 表関数

モニター・レベル	SQL 表関数	戻される情報
データベース・マネージャー	HEALTH_DBM_INFO	データベース・マネージャー・レベルからのヘルス・スナップショットに関する基本情報
データベース・マネージャー	HEALTH_DBM_HI	データベース・マネージャー・レベルからのヘルス・インディケーター情報
データベース・マネージャー	HEALTH_DBM_HI_HIS	データベース・マネージャー・レベルからのヘルス・インディケーター履歴情報
データベース	HEALTH_DB_INFO	データベースからのヘルス・スナップショットに関する基本情報
データベース	HEALTH_DB_HI	データベースからのヘルス・インディケーター情報
データベース	HEALTH_DB_HI_HIS	データベースからのヘルス・インディケーター履歴情報
データベース	HEALTH_DB_HIC	データベースのコレクション・ヘルス・インディケーターに関するコレクション情報
データベース	HEALTH_DB_HIC_HIS	データベースのコレクション・ヘルス・インディケーターに関するコレクション履歴情報
表スペース	HEALTH_TBS_INFO	データベースの表スペースのヘルス・スナップショットに関する基本情報
表スペース	HEALTH_TBS_HI	データベースの表スペースに関するヘルス・インディケーター情報
表スペース	HEALTH_TBS_HI_HIS	データベースの表スペースに関するヘルス・インディケーター履歴情報
表スペース	HEALTH_CONT_INFO	データベースのコンテナーのヘルス・スナップショットに関する基本情報
表スペース	HEALTH_CONT_HI	データベースのコンテナーに関するヘルス・インディケーター情報
表スペース	HEALTH_CONT_HI_HIS	データベースのコンテナーに関するヘルス・インディケーター履歴情報

ヘルス・モニター CLP コマンド:
次の表に、サポートされているスナップショット要求のタイプをすべてリストします。

表 67. スナップショット・モニター CLP コマンド

モニター・レベル	CLP コマンド	戻される情報
データベース・マネージャー	`get health snapshot for dbm`	データベース・マネージャー・レベル情報。
データベース	`get health snapshot for all databases`	データベース・レベル情報。情報が返されるのは、データベースがアクティブ化されている場合に限られます。

表 67. スナップショット・モニター *CLP* コマンド *(続き)*

モニター・レベル	CLP コマンド	戻される情報
データベース	`get health snapshot for database on database-alias`	データベース・レベル情報。情報が返されるのは、データベースがアクティブ化されている場合に限られます。
データベース	`get health snapshot for all on database-alias`	データベース、表スペース、および表スペース・コンテナーの情報。情報が返されるのは、データベースがアクティブ化されている場合に限られます。
表スペース	`get snapshot for tablespaces on database-alias`	データベースに接続されたアプリケーションがアクセスしている各表スペースの表スペース・レベルの情報。また、表スペース中の各表スペース・コンテナーに関する正常性の情報も含みます。

ヘルス・モニター API 要求タイプ:

次の表に、サポートされているスナップショット要求のタイプをすべてリストします。

表 68. スナップショット・モニター *API* 要求タイプ

モニター・レベル	API 要求タイプ	戻される情報
データベース・マネージャー	SQLMA_DB2	データベース・マネージャー・レベル情報。
データベース	SQLMA_DBASE_ALL	データベース・レベル情報。情報が返されるのは、データベースがアクティブ化されている場合に限られます。
データベース	SQLMA_DBASE	データベース・レベル情報。情報が返されるのは、データベースがアクティブ化されている場合に限られます。
表スペース	SQLMA_DBASE_TABLESPACES	データベースに接続されたアプリケーションがアクセスしている各表スペースの表スペース・レベルの情報。また、表スペース中の各表スペース・コンテナーに関する正常性の情報も含みます。

メモリー・ビジュアライザーでの作業

メモリー・ビジュアライザーは、インスタンスおよびそのすべてのデータベースのメモリー関連のパフォーマンスを、データベース管理者がモニターするのに役立ちます。階層ツリーに編成されたメモリー・コンポーネントのメモリー使用率の最新の情報を視覚的に表示することができます。

重要: メモリー・ビジュアライザーは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。 詳しくは、「*DB2 バージョン 9.7 の新機能*」のトピック『コントロール・センター・ツールおよび DB2 管理サーバー (DAS) が推奨されなくなった』を参照してください。

メモリー・パフォーマンスと使用量のプロットを表示したり、メモリー・ビジュアライザーの構成パラメーターを更新したりするには、SYSADM 権限がなければなりません。

メモリー・ビジュアライザーを使用してパフォーマンス上の問題をトラブルシューティングすることができます。メモリー・コンポーネントの構成パラメーターの設定を変更して、変更による効果を評価することができます。メモリーは必要に応じて割り振られるため、構成パラメーターは DB2 でのメモリー使用量に影響を与えます。構成パラメーターの値を許容範囲よりも大きくまたは小さく設定した場合、エラー・メッセージが表示されます。構成パラメーターを変更するとメモリー・ビジュアライザー内に即時に影響を与え、新しい値は次のリフレッシュ・サイクルの時に組み込まれます。

- メモリー・ビジュアライザーを使用してメモリー・パフォーマンスを表示する方法
  1. Windows の「**スタート**」メニューから、「**プログラム**」→「**IBM DB2**」→「**モニター (Monitoring)**」→「**ツール (Tools)**」→「**メモリー・ビジュアライザー (Memory Visualizer)**」とクリックして、メモリー・ビジュアライザーを開きます。「メモリー・ビジュアライザー・インスタンスの選択 (Memory Visualizer instance selection)」ウィンドウが開きます。「**インスタンス名**」フィールドからインスタンスを選択し、「**OK**」をクリックします。
  2. データベースおよびそれらの関連したメモリー・コンポーネントが階層ツリーに表示されるまで、インスタンス・オブジェクト・ツリーを展開します。メモリー・プールの値が、メモリー・ビジュアライザー・ウィンドウに表示されます。
  3. メモリー・コンポーネントのプロットされたグラフを表示するには、以下のいずれかの方法を使用します。
     - 階層ツリーでコンポーネントを選択し、メモリー・ビジュアライザー・ウィンドウの「**プロットの表示**」チェック・ボックスをクリックする。
     - 選択済みメモリー・コンポーネントを右クリックしてポップアップ・メニューを表示し、「**プロットの表示**」を選択する。
     - 階層ツリーでコンポーネントを選択し、ツールバー上の「選択」メニューから「**プロットの表示**」オプションを選択する。各メモリー・コンポーネントのプロットされたデータが「メモリー使用率プロット」に表示されます。
     - 別のメモリー・コンポーネントのデータを表示する場合、それを階層ツリーで選択し、「**プロットの表示**」チェック・ボックスをクリックする。コンポーネントのプロットされたデータが、他のコンポーネントと一緒に「メモリー使用率プロット」に表示されます。

     グラフには、時間の経過とともに収集されたメモリー・コンポーネントのデータが表示されます。各コンポーネントは、メモリー・ビジュアライザー・ウィンドウの「**プロットの凡例**」フィールドに表示されているものと同じ色と形状で示されます。その形状は間隔を置いて繰り返し表示されます。グラフのプロットのラベルは、コンポーネントを示します。

     パフォーマンス・データがキャプチャーされた時間は、グラフの下に表示されます。グラフの時間間隔は変更できます。

注: プロットに新しいメモリー・コンポーネントを追加しても、前に追加されたメモリー・コンポーネントは置き換わりません。

水平および垂直スクロール・バーにより、プロットされたデータの異なるビューを表示することができます。

- グラフの下部にある水平スクロール・バーを使用すると、選択した時間枠におけるメモリー・コンポーネントの履歴データを表示できます。スライダー・バーをポイントして、グラフの底に沿ってドラッグします。
- グラフの右方にある垂直スクロール・バーを使用すると、選択したコンポーネントのメモリー使用率を表示できます。ビューを変更するには、スライダーをポイントしてドラッグします。

メモリー使用率がそれまでの最高に達すると、垂直スクロール・バーの最大値は更新されて新しい値が反映されます。垂直スクロール・バーの最小値を 0 以外の値に設定して、異なる範囲のプール使用率値を表示できます。

- メモリー・ビジュアライザー・データ・ファイルのデータを、新しいメモリー・ビジュアライザー・ウィンドウにロードできます。このデータは、インスタンスおよびそのすべてのデータベースのパフォーマンスと、履歴データを比較するために使用できます。メモリー・ビジュアライザー・データ・ファイルからデータをロードするには、メモリー・ビジュアライザーのメニューから「**オープン**」を選択し、その後「オープン」ダイアログで拡張子 *.mdf を持つデータ・ファイルを選択します。
- 「**時刻単位**」フィールドを使用すると、「メモリー使用率プロット」ウィンドウの時間間隔を変更できます。グラフ・データの時間間隔のデフォルトは、分です。間隔は、分、時、または日を選択できます。時間間隔を選択すると、新しい時間間隔がグラフの水平範囲に表示され、水平スクロール・バーの段階移動量が変更されます。
- 「メモリー使用率プロット」からメモリー・コンポーネントのプロットされたグラフを除去するには、階層ツリーでコンポーネントを選択してメモリー・ビジュアライザー・ウィンドウの「**プロットの表示**」チェック・ボックスのチェックを外すか、または選択済みメモリー・コンポーネントを右クリックしてポップアップ・メニューを表示し、「**プロットの表示**」を選択解除します。コンポーネントのプロットされたデータは、「メモリー使用率プロット」ウィンドウから除去されます。コンポーネントを示していた色と形状は、メモリー・ビジュアライザー・ウィンドウの「**プロットの凡例**」フィールドに表示されなくなります。
- メモリー・パフォーマンスの履歴の追跡および作成に役立つように、メモリー・ビジュアライザーの実行中にメモリー・パフォーマンス・データ (プロットされたグラフを含む) を保管できます。メモリー・パフォーマンス・データを保管するには、メモリー・ビジュアライザーのメニューから「**保管**」または「**別名保管**」を選択した後、ファイルのロケーションと、拡張子 `.mdf` を持つファイル名を選択します。
- メモリー・コンポーネントの構成パラメーターの設定を変更する方法
  1. 必要なメモリー・プールを展開してその構成パラメーターが階層ツリーに表示されるようにします。
  2. コンポーネントをクリックして選択し、「**パラメーター値**」列でその数値をクリックします。テキスト・ボックスには、コンポーネントの現行値が表示されます。テキスト・ボックスに新しい数値を入力し、**Enter** を押します。新しい

値は、構成パラメーターが更新されるまで (これはおそらく次のリフレッシュ・サイクルで行われます)、「**パラメーター値**」列の元の値の横に表示されます。さらに、選択したコンポーネントの「**パラメーター値**」列の値を右クリックして、ポップアップ・メニューを表示することもできます。列の外側をクリックして、変更を完了します。 メモリー・コンポーネントの新しい値は、「**パラメーター値**」列の元の値の横に表示されます。メモリー・パフォーマンスのグラフを表示する選択をした場合、グラフのプロット・ビューに新しい値が表示されます。この変更はメモリー・ビジュアライザーでは即時に行われますが、DB2 内の構成パラメーターに対して行う変更の更新には遅延があります。構成パラメーターの値は、ポップアップ・メニューの「**デフォルトにリセット**」オプションを使用してリセットできます。

メモリー・ビジュアライザーの概要

メモリー・ビジュアライザーを使用して、特定のインスタンスとそのすべてのデータベースのメモリーに関連したパフォーマンスをモニターします。

重要: メモリー・ビジュアライザーは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。 詳しくは、「*DB2 バージョン 9.7 の新機能*」のトピック『コントロール・センター・ツールおよび DB2 管理サーバー (DAS) が推奨されなくなった』を参照してください。

メモリー・ビジュアライザーをオープンし、階層ツリーで 1 つまたは複数のメモリー・コンポーネントを選択します。「メモリー・ビジュアライザー (Memory Visualizer)」ウィンドウに、コンポーネントに割り振られるメモリー量と現行のメモリー使用量の値が表示されます。「メモリー・ビジュアライザー (Memory Visualizer)」ウィンドウに、ツリー・ビューと履歴ビューの 2 つのデータ・ビューが表示されます。一連の列には、アラームと警告の上限および下限に関するしきい値 (パーセンテージ) が示されます。また、列にはリアルタイムのメモリー使用率も表示されます。

注: バージョン 8.1 以降のインスタンスについては、メモリー・ビジュアライザーを使用してメモリー・パフォーマンス・データを提供することができます。

以下のリストは、メモリー・ビジュアライザーを使って行える主要なタスクのいくつかを分類しています。

- DB2 インスタンスとそのデータベースについて、選択コンポーネントのメモリー使用率に関するさまざまな列のデータを表示または非表示にする。
- メモリー・パフォーマンス・データのグラフを表示する。
- 構成パラメーターを更新することにより、個々のメモリー・コンポーネントの設定を変更する。
- ファイルから「メモリー・ビジュアライザー (Memory Visualizer)」ウィンドウにパフォーマンス・データをロードする。
- メモリー・パフォーマンス・データを保管する。

メモリー・ビジュアライザー・インターフェースには、特定のインスタンスとそのすべてのデータベースのメモリーに関連したパフォーマンスをモニターする上で役に立つ次のエレメントがあります。

メモリー・ビジュアライザー・ウィンドウ

「メモリー・ビジュアライザー (Memory Visualizer)」ウィンドウの列には、メモリー・コンポーネントのパフォーマンスの値が表示されます。以下の情報が表示されます。

プロットの凡例
「メモリー使用量のプロット (Memory Usage Plot)」に表示されるチェック済みメモリー・コンポーネントまたは構成パラメーター。周期的にプロット・グラフに現れる特定の形状がそれぞれのコンポーネントまたはパラメーターを識別します。

使用率 データベース・オブジェクトに割り振られ、そこで使用されるメモリーのサイズ。使用率と構成済み割り振りを示したグラフィカル・バーが含まれます。バーの長さは一定で、埋められた部分が使用率を示します (パーセンテージ)。

パラメーター値
構成パラメーターの現行値。

上限アラーム (%) しきい値
上限アラームを生成するしきい値。デフォルト値は 98% です。

上限警告 (%) しきい値
上限警告を生成するしきい値。デフォルト値は 90% です。

下限アラーム (%) しきい値
下限アラームを生成するしきい値。デフォルト値は 2% です。

下限警告 (%) しきい値
下限警告を生成するしきい値。デフォルト値は 10% です。

使用量を示すグラフィカル・バー

「メモリー・ビジュアライザー (Memory Visualizer)」ウィンドウ内にある、使用量を示すグラフィカル・バーは、メモリー使用率の視覚的合図です。選択されたメモリー・コンポーネントによって使用されているメモリー量、およびその使用量がシステムに与え得る影響を判断する上でこのバーは役に立ちます。メモリー・ビジュアライザーは、使用量に対応する値をパーセンテージで表したものも表示します。この 2 つの標識は、コンポーネントの構成パラメーターの設定を変更する必要があるか、あるいは別の適切なアクションを取る必要があるかを判断する上で役に立ちます。

メモリー・コンポーネント

データベース・マネージャーはシステムでさまざまな種類のメモリー、つまり、データベース・マネージャー共用メモリー、データベース・グローバル・メモリー、アプリケーション・グローバル・メモリー、エージェント/アプリケーション共用メモリー、エージェント専用メモリーを使用します。これらの種類のメモリーは、展開する階層ツリーの編成でメモリー・ビジュアライザーが使用する上位のメモリー・コンポーネントです。

それぞれの上位のメモリー・コンポーネントの基礎には、メモリーの割り振りおよび割り振り解除の方法を決定する他のコンポーネントがあります。例えば、メモリーの割り振りおよび割り振り解除は、データベース・マネージャーの開始時、データベースの活動化時、アプリケーションのデータベースへの接続時、またエージェントのアプリケーション作業への割り当て時に行

われます。メモリー・ビジュアライザーはこれらのリーフ・レベルのメモリー・コンポーネントを使用して、メモリーが DB2 インスタンスでどのように割り振られ、使用されるかを表示します。

階層ツリーの編成

メモリー・ビジュアライザーは階層ツリーの編成を使用しますが、これは DB2 でメモリー・コンポーネントを表示し、ブラウズする上で助けになります。階層ツリーでは、個々のメモリー・コンポーネントの情報を展開し、列、グラフィカルな表示、およびグラフによって表示することができます。

ツリー・ビューは主に、4 種類のメモリー項目で構成されます。

DB2 インスタンス
現在システムで実行されているインスタンス。

データベース
インスタンス上で定義されているデータベース。

上位のメモリー・コンポーネント
リーフ・レベルのメモリー・コンポーネントの論理的なグループ。例えば、データベース・マネージャー共用メモリー、データベース・グローバル・メモリー、エージェント専用メモリー、エージェント/アプリケーション共用メモリーなどにグループ化できます。

リーフ・レベルのメモリー・コンポーネント
「メモリー・ビジュアライザー (Memory Visualizer)」ウィンドウに表示されるメモリー・コンポーネント。例えば、バッファー・プール、ソート・ヒープ、データベース・ヒープ、ロック・リストなどがあります。

次のツリー・ビューのアイコンはそれぞれメモリー・ツリーの項目を表します。

- インスタンス: 
- データベース: 
- 上位メモリーのグループ: 
- リーフ・レベルのメモリー・コンポーネント: 

ツリー項目のメモリー使用率がしきい値を超えると、色付きの標識がアイコン上に表示されます。黄色は警告状態を示します。赤色はアラーム状態を示します。

履歴ビューには、ツリー・ビューで選択されたメモリー・コンポーネントのデータが表示されます。データには、割り振りメモリーおよび使用メモリーの値、プロット・グラフ、およびメモリー・ビジュアライザーの実行中に構成パラメーターに対して行われた変更が含まれます。データは一定期間メモリー・ビジュアライザーに保管されます。メモリー・ビジュアライザーのデータ・ファイルにメモリー・パフォーマンス・データを保管して、トラッキングや他のデータとの比較、またはトラブルシューティングなどのために使用できます。

メモリー使用量のグラフ

メモリー使用量のグラフには、「メモリー使用量のプロット (Memory Usage Plot)」で選択されたメモリー・コンポーネントのプロット・データが表示されます。グラフ内の各コンポーネントは特定の色で識別され、これは「メモリー・ビジュアライザー (Memory Visualizer)」ウィンドウの「プロットの凡例 (Plot Legend)」列にも表示されます。グラフには構成パラメーターの設定に対して行われた変更も表示されます。構成パラメーターの元の値と新しい値の設定、および変更の要求時刻がグラフに表示されます。これらは履歴ビューに組み込まれ、このビューを使用してメモリー・パフォーマンスの評価を行えます。

詳しくは、 209 ページの『メモリー・ビジュアライザーでの作業』 を参照してください。

アクティビティー・モニターの概要

アクティビティー・モニターを使用して、データベースまたはデータベース・パーティションのアプリケーションのパフォーマンスと並行性、リソース消費量、および SQL ステートメントの使用状況をモニターします。

重要: アクティビティー・モニターは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。 詳しくは、「*DB2 バージョン 9.7 の新機能*」のトピック『コントロール・センター・ツールおよび DB2 管理サーバー (DAS) が推奨されなくなった』を参照してください。

アクティビティー・モニターは、特定のモニター・データのサブセットに基づく事前定義のレポートのセットを提供します。これらのレポートによって、モニター対象をアプリケーションのパフォーマンス、アプリケーションの並行性、リソース消費量、および SQL ステートメントの使用状況に絞ることができます。また、アクティビティー・モニターは、ほとんどのレポートに関する推奨を提供します。この推奨は、データベースのパフォーマンス上の問題の原因を診断し、データベース・リソースを最も有効利用できるように照会を調整する上で役に立ちます。

216 ページの図 7 では、アクティビティー・モニターを使用して問題を解決するためのプロセスを説明しています。

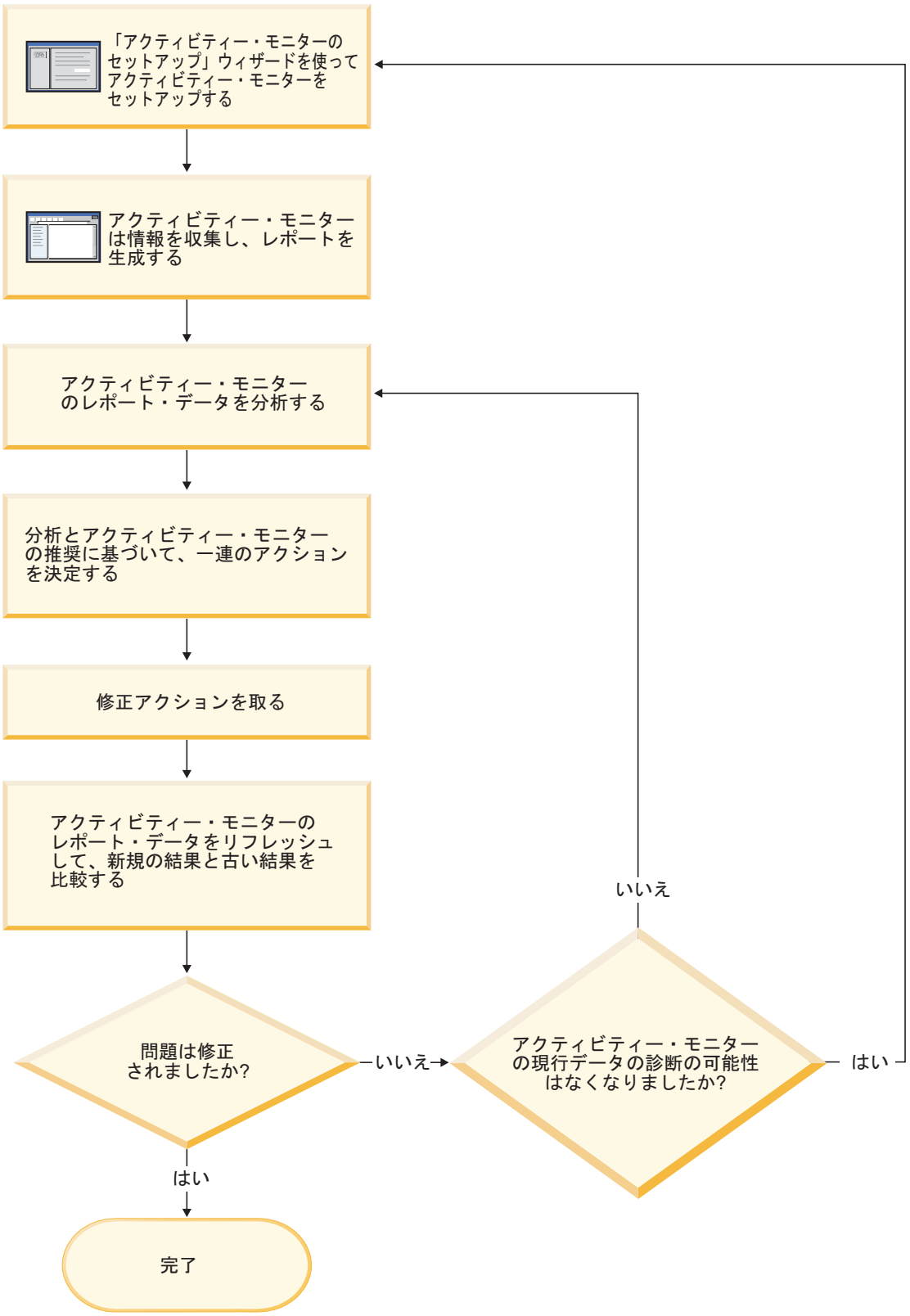


図 7. アクティビティー・モニターの概要

表 69. アクティビティー・モニターから実行できるタスク

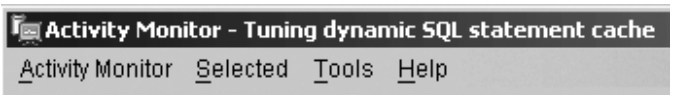
アクティビティー・モニターからのタスク	**タスクの特徴**	**呼び出し方法**
トランザクション	選択されたアプリケーションで実行されているトランザクションを表示します。	「**レポート・データ**」ペインで 1 つ以上のアプリケーションを選択します。右クリックし、「**最後のトランザクションの表示**」を選択します。「アプリケーション・トランザクション」ウィンドウが開きます。
ステートメント	選択されたアプリケーションで実行されている SQL ステートメントを表示します。	「**レポート・データ**」ペインで 1 つ以上のアプリケーションを選択します。右クリックし、「**最後のステートメントの表示**」を選択します。「アプリケーション・ステートメント」ウィンドウが開きます。
	選択されたアプリケーションで実行されている SQL ステートメントのテキストを表示します。	「アプリケーション・ステートメント」ウィンドウの「**レポート・データ**」ペインで、1 つのステートメントを右クリックします。「**ステートメント・テキストの表示**」を選択します。
アプリケーション・ロック・チェーン	選択されているアプリケーションに現在影響を与えるロックおよびロック待機状態を表示します。	「**レポート・データ**」ペインでアプリケーションを選択します。右クリックし、「**ロック・チェーンの表示**」を選択します。「アプリケーション・ロック・チェーン」ウィンドウが開きます。
	ロック情報を表示している選択アプリケーションに関する情報を表示します。	「アプリケーション・ロック・チェーン」ウィンドウからアプリケーションを右クリックし、「**情報**」を選択します。
	データベース内で選択されているアプリケーションによって保留されているロックおよび待機されているロックに関する情報を表示します。	「アプリケーション・ロック・チェーン」ウィンドウからアプリケーションを右クリックし、「**ロックの詳細の表示**」を選択します。

表 69. アクティビティー・モニターから実行できるタスク (続き)

アクティビティー・モニターからのタスク	タスクの特徴	呼び出し方法
レポート・データおよび推奨の表示	レポート・データを解釈する上で役に立つ情報を表示します。	「アクティビティー・モニター」ウィンドウ、「アプリケーション・ステートメント」ウィンドウ、または「アプリケーション・トランザクション」ウィンドウから、「**レポート**」矢印を使用してレポートを選択し、「**レポートの詳細 (Report Details)**」プッシュボタンをクリックします。「**詳細**」ページを表示します。
	アクティビティー・モニターによって提供される推奨を表示します。	「アクティビティー・モニター」ウィンドウ、「アプリケーション・ステートメント」ウィンドウ、または「アプリケーション・トランザクション」ウィンドウから、「**レポート**」矢印を使用してレポートを選択し、「**レポートの詳細 (Report Details)**」プッシュボタンをクリックします。「**推奨**」ページを表示します。

アクティビティー・モニターのインターフェースには、収集されるモニター・データを編成し、解釈する上で役に立ついくつかのエレメントがあります。

メニュー・バー

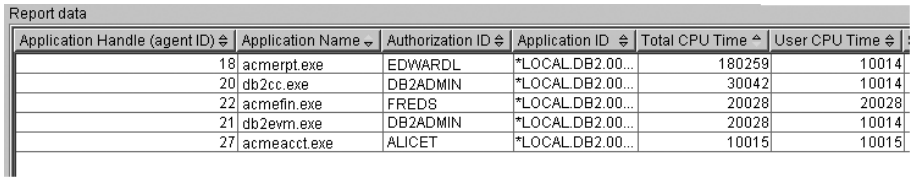


メニュー・バーを使用して、アクティビティー・モニターのオブジェクトを処理し、別の管理センターおよびツールを開いて、オンライン・ヘルプにアクセスします。

アクティビティー・モニターのツールバー

ツールバー・アイコンを使用して DB2 ツールを開き、DB2 情報を表示します。

レポート・データ・ペイン



「**レポート・データ**」ペインを使用して、アクティビティー・モニターで使用可能なレポート・データを表示し、それを処理します。「**レポート・データ**」ペインには、「**レポート**」フィールドで選択されるレポートの内容を構成する項目が表示されます。

また、「**レポート・データ**」ペインでは、他の「アクティビティー・モニター」ウィンドウへのアクセスも提供されます。アクティビティー・モニターでは、モニター操作を行っているアプリケーションから、そのアプリケーションが実行している個々のトランザクションまたは個々の SQL ステートメントにドリルダウンすることができます。

レポート・データ・ペインのツールバー

「**レポート・データ**」ペインの下にあるツールバーを使用して、オブジェクトのビューや「**レポート・データ**」ペインの情報を必要に応じて調整することができます。

モニターのシナリオ

シナリオ: スナップショット管理ビューを使用してコストの高いアプリケーションを識別する

最近、ShopMart データベース上のワークロードが増えたために、データベース全体のパフォーマンスが低下し始めています。 ShopMart DBA の Jessie は、次の管理ビューを使って、日々のワークロードの中からとりわけリソースを消費しているものを識別しようとしています。

APPLICATION_PERFORMANCE

このビューは、頻繁に表スキャンを実行している可能性のあるアプリケーションを Jessie が識別するのに役に立ちます。

```
connect to shopmart;
select AGENT_ID, ROWS_SELECTED, ROWS_READ from APPLICATION_PERFORMANCE;
```

ROWS_SELECTED の値は、アプリケーションに戻される行数を示し、ROWS_READ の値は、基本表からアクセスされる行数を示します。選択度が低い場合、アプリケーションは、索引の作成によって避けられるかもしれない表スキャンを実行している可能性があります。 Jessie はこのビューを使用して、問題が発生する可能性のある照会を識別し、その後 SQL を調べることによって詳細な調査を行い、照会の実行時に読み取られる行数を減らす方法がないか調べることができます。

LONG_RUNNING_SQL

Jessie は LONG_RUNNING_SQL 管理ビューを使用して、現在最も長い時間実行されている照会を識別します。

```
connect to shopmart;
select ELAPSED_TIME_MIN, APPL_STATUS, AGENT_ID
 from long_running_sql order by ELAPSED_TIME_MIN desc
 fetch first 5 rows only;
```

このビューを使用すると、これらの照会の実行時間、および照会の状況を判別することができます。照会が長時間実行されており、ロックを待機している場合は、LOCKWAITS または LOCK_HELD 管理ビューを使用して特定のエージェント ID を照会し、詳細を調査できます。
LONG_RUNNING_SQL ビューからも実行中のステートメントが分かるので、問題が発生する可能性のある SQL を識別することができます。

QUERY_PREP_COST

Jessie は QUERY_PREP_COST を使用して、問題ありと確認された照会のトラブルシューティングを行うことができます。このビューから、照会の実行頻度、および各照会の平均実行時間が分かります。

```
connect to shopmart;
select NUM_EXECUTIONS, AVERAGE_EXECUTION_TIME_S, PREP_TIME_PERCENT
 from QUERY_PREP_COST  order by NUM_EXECUTIONS desc;
```

PREP_TIME_PERCENT の値から、Jessie は、照会実行時間のうち何パーセントが照会の準備に費やされているかが分かります。照会のコンパイルと最適化にかかる時間が照会の実行にかかる時間とほぼ同じである場合、Jessie は照会の所有者に、照会に使用する最適化クラスの変更を勧めることもできます。最適化クラスを小さくすることにより照会は最適化をより素早く完了し、結果をより短時間で戻せるかもしれません。しかし、照会の準備に長時間がかかるとは言え、(再準備なしで) 何千回も実行されるのであれば、最適化クラスを変更しても照会パフォーマンスは向上しないかもしれません。

TOP_DYNAMIC_SQL

Jessie は TOP_DYNAMIC_SQL ビューを使用して、最も頻繁に実行され、最も長い時間実行され、そして最も頻繁にソートが行われる動的 SQL ステートメントを識別します。この情報があれば、Jessie は SQL の調整の努力を、最もリソースを消費するいくつかの照会に集中することができます。

最も頻繁に実行される動的 SQL ステートメントを識別するために、Jessie は次のコマンドを発行します。

```
connect to shopmart;
select * from TOP_DYNAMIC_SQL order by NUM_EXECUTIONS desc
fetch first 5 rows only;
```

これは、実行時間、実行されるソートの数、および最も頻繁に実行される動的 SQL ステートメントの上位 5 つのステートメント・テキストに関するすべての詳細を戻します。

実行時間が最も長い動的 SQL ステートメントを識別するために、Jessie は、AVERAGE_EXECUTION_TIME_S の値が上位 5 つの照会を調べます。

```
connect to shopmart;
select * from TOP_DYNAMIC_SQL
 order by AVERAGE_EXECUTION_TIME_S desc fetch first 5 rows only;
```

最も頻繁にソートが行われる動的 SQL ステートメントの詳細を見るために、Jessie は次のコマンドを発行します。

```
connect to shopmart;
select STMT_SORTS, SORTS_PER_EXECUTION, substr(STMT_TEXT,1,60) as STMT_TEXT
 from TOP_DYNAMIC_SQL order by STMT_SORTS desc
 fetch first 5 rows only;
```

シナリオ: 管理ビューを使用したバッファー・プール効率のモニター

DBA の John は、SALES データベースのアプリケーション・パフォーマンスの低下の原因は、バッファー・プールが効率的に機能していないことにあると考えています。これを調査するため、BP_HITRATIO 管理ビューを使ってバッファー・プールのヒット率を調べます。

```
connect to SALES;
select BPNAME, TOTAL_HIT_RATIO from BP_HIT_RATIO;
```

John は、あるバッファー・プールのヒット率が非常に低いことに気がつきます。これは、大量のページがバッファー・プールからではなく、ディスクから読み取られるということです。

そこで、BP_READ_IO 管理ビューを使って、プリフェッチャーの調整が必要かどうかを調べます。

```
connect to SALES;
select BPNAME, PERCENT_SYNC_READS,UNUSED_ASYNC_READS_PERCENT from BP_READ_IO;
```

PERCENT_SYNC_READS の値から、プリフェッチなしで同期的に読み取られるページのパーセンテージが分かります。数値が高いなら、それはディスクから直接読み取られているデータのパーセンテージが高いということです。プリフェッチャーがさらに必要であることを示しているのかもしれません。
UNUSED_ASYNC_READS_PERCENT の値から、ディスクからの読み取りは非同期に行われたものの、照会でまったくアクセスされていないページのパーセンテージが分かります。これは、プリフェッチャーが過剰にデータ・ページを読み取っているために I/O が不必要に生じていることを示しているのかもしれません。

PERCENT_SYNC_READS の値と UNUSED_ASYNC_READS_PERCENT の値のどちらも許容範囲内にあると思われるため、John は BP_WRITE_IO 管理ビューを使用して、ページ・クリーナーがどの程度読み込まれるデータページのために既存ページスペースをクリアしているかを調査します。

```
connect to SALES;
select BPNAME, PERCENT_WRITES_ASYNC from BP_WRITE_IO;
```

PERCENT_WRITES_ASYNC の値から、John は、物理書き込み要求の何パーセントが非同期的に実行されたかが分かります。この数値が高い場合は、ページ・クリーナーがよく機能しており、新しいデータ・ページ要求の着信前にバッファー・プール内のスペースを消去しているということかもしれません。この数値が低い場合は、データ・ページがバッファー・プールに読み込まれるのをアプリケーションが待っている間に、データベース・エージェントが実行する物理書き込みの数が増えるということです。

John は PERCENT_WRITES_ASYNC の値が 25% と非常に低いことに気がつきます。そこで、非同期書き込みの比率を上げるため、SALES データベースに対してさ

らに多くのページ・クリーナーを構成することにします。ページ・クリーナーの数を増やした後、再びバッファー・プール管理ビューを使って、チューニングの効果を確かめることができます。

アクティビティー・モニターのセットアップ

アプリケーションのパフォーマンスと並行性、リソース使用量、データベースまたはデータベース・パーティションでの SQL ステートメントの使用をモニターするために、アクティビティー・モニターをセットアップできます。アクティビティー・モニターは、特定のモニター・データのサブセットに基づく事前定義のレポートのセットを提供します。アクティビティー・モニターはさらに、データベースのパフォーマンス上の問題の原因の診断を支援したり、データベース・リソースの使用が最適になるように照会を調整したりするための推奨を提供することができます。

重要: アクティビティー・モニターは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。 詳しくは、「*DB2 バージョン 9.7 の新機能*」のトピック『コントロール・センター・ツールおよび DB2 管理サーバー (DAS) が推奨されなくなった』を参照してください。

アクティビティー・モニターを使用するには、以下のようにします。

- サーバーには DB2 UDB バージョン 8.2 以降が必要です。
- SQLADM または DBADM 権限が必要です。

「アクティビティー・モニターのセットアップ」ウィザードをオープンします。

- コントロール・センターから、アクティビティー・モニターをセットアップする対象のインスタンスまたはデータベースが見つかるまでオブジェクト・ツリーを展開します。オブジェクトを右クリックして、ポップアップ・メニューから「アクティビティー・モニターのセットアップ」を選択します。
- コマンド行から、db2am コマンドを入力してください。

詳細情報は、コントロール・センター内のコンテキスト・ヘルプ機能を使用して入手できます。

ロールバック・プロセスの進捗モニター

トランザクションのロールバック中にアプリケーションのスナップショットを取得すると、出力にロールバック・モニター・エレメントが表示されます。この情報は、ロールバック操作の進捗状況をモニターするのに使用できます。

アプリケーション・スナップショットでは、ロールバックの開始時刻、行われる作業の全体量、およびすでに完了した作業量についての情報が提供されます。作業はバイト数で測定されます。

以下は、GET SNAPSHOT FOR ALL APPLICATIONS コマンドによる出力の例です。

```
            Application Snapshot

Application handle         = 6
Application status         = Rollback Active
      Start Time           = 02/20/2004 12:49:27.713720
```

```
      Completed Work       = 1024000 bytes
      Total Work           = 4084000 bytes

            Application Snapshot

Application handle         = 10
Application status         = Rollback to Savepoint
      Start Time           = 02/20/2004 12:49:32.832410
      Completed Work       = 102400 bytes
      Total Work           = 2048000 bytes
```

「アプリケーション状況」モニター・エレメントの値は、どのタイプのロールバック・イベントが発生しているかを示します。それぞれの値の意味は次のとおりです。

ロールバック・アクティブ
: これは、ロールバックの作業単位で、トランザクション全体の明示的な (ユーザー呼び出し)、または暗黙の (強制) ロールバックが含まれます。

セーブポイント・ロールバック
: これは、ステートメントまたはアプリケーション・レベルのセーブポイントまでの、部分的なロールバックです。ネストされたセーブポイントは単一の単位と見なされ、最外部のセーブポイントが使用されます。

完了作業単位は、ロールバックが完了した、ログ・ストリーム内の相対位置を示します。完了作業に対する更新は、ログ・レコードが処理されるたびに行われますが、更新の実行は、ログ・レコードのサイズが異なるため等間隔では行われません。

合計作業単位は、トランザクションまたはセーブポイントでロールバックされる必要のある、ログ・ストリームにあるログ・レコードの範囲に基づいた見積量です。これは、処理が必要なログ・レコードの正確なバイト数を示したものではありません。

スナップショット・モニター・データを使用したパーティション表の再編成のモニター

以下の情報は、表再編成の全体的な状況をモニターするための便利な方法のいくつかを説明しています。

パーティション表の表の再編成の状況全体を示す、別個のデータ・グループはありません。パーティション表は、データ・パーティションまたは範囲と呼ばれる複数のストレージ・オブジェクトに表データを分割するというデータ編成スキームを使用します。分割は、表の 1 つ以上の表パーティション・キー列の値に従って行われます。しかし、再編成中の個々のデータ・パーティションのデータ・グループ内のエレメントの値から、表再編成の全体的な状況を推察することができます。以下の情報は、表再編成の全体的な状況をモニターするための便利な方法のいくつかを説明しています。

再編成中のデータ・パーティションの数の判別

1 つの表の再編成中のデータ・パーティションの総数は、表名とスキーマ名が同じ表データのモニター・データ・ブロックの数を数えることで判別できます。この値

は再編成が開始したデータ・パーティションの数を示します。例 1 と 2 は、3 つのデータ・パーティションが再編成中であることを示します。

再編成中のデータ・パーティションの識別

フェーズの開始時刻 (reorg_phase_start) から、再編成中の現行データ・パーティションを推察することができます。 SORT/BUILD/REPLACE フェーズの間は、再編成中のデータ・パーティションに対応するモニター・データが最新のフェーズ開始時刻を示します。 INDEX_RECREATE フェーズの間は、データ・パーティションのフェーズの開始時刻がすべて同じになります。例 1 と 2 では INDEX_RECREATE フェーズが示されており、すべてのデータ・パーティションの開始時刻が同じになっています。

索引再ビルド要件の識別

再編成中のいずれか 1 つのデータ・パーティションと対応する最大再編成フェーズ・エレメント (reorg_max_phase) の値を入手することにより、索引の再ビルドが必要かどうかを判別することができます。 reorg_max_phase の値が 3 または 4 の場合、索引の再ビルドが必要になります。例 1 および 2 では reorg_max_phase が 3 と報告されており、これは索引の再ビルドが必要であるということを意味しています。

以下の出力例は、3 つのデータ・パーティションを持つ 1 つの表を含んだ、3 ノードで構成されているサーバーからのものです。

```
CREATE TABLE  sales (c1 INT, c2 INT, c3 INT)
    PARTITION BY RANGE (c1)
        (PART P1 STARTING FROM (1)  ENDING AT (10) IN parttbs,
         PART P2 STARTING FROM (11) ENDING AT (20) IN parttbs,
         PART P3 STARTING FROM (21) ENDING AT (30) IN parttbs)
    DISTRIBUTE BY (c2)
```

実行されるステートメント:

```
REORG TABLE sales ALLOW NO ACCESS ON ALL DBPARTITIONNUMS
```

例 1:

```
GET SNAPSHOT FOR TABLES ON DPARTDB GLOBAL
```

出力は関係のある表の情報だけを含むように変更されています。

```
           Table Snapshot

First database connect timestamp     = 06/28/2005 13:46:43.061690
Last reset timestamp                 = 06/28/2005 13:46:47.440046
Snapshot timestamp                   = 06/28/2005 13:46:50.964033
Database name                        = DPARTDB
Database path                        = /work/sales/NODE0000/SQL00001/
Input database alias                 = DPARTDB
Number of accessed tables            = 5

Table List
 Table Schema        = NEWTON
 Table Name          = SALES
 Table Type          = User
 Data Partition Id   = 0
 Data Object Pages   = 3
 Rows Read           = 12
```

```
Rows Written        = 1
Overflows           = 0
Page Reorgs         = 0
Table Reorg Information:
  Node number       = 0
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:49.816883
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.362918
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.821244

Table Reorg Information:
  Node number       = 1
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:49.822701
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.420741
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.899543


Table Reorg Information:
  Node number       = 2
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:49.814813
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.344277
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.803619
```

```
Table Schema        = NEWTON
Table Name          = SALES
Table Type          = User
Data Partition Id   = 1
Data Object Pages   = 3
Rows Read           = 8
Rows Written        = 1
Overflows           = 0
Page Reorgs         = 0
Table Reorg Information:
  Node number       = 0
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.014617
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.362918
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.821244

Table Reorg Information:
  Node number       = 1
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.026278
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.420741
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.899543

Table Reorg Information:
  Node number       = 2
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.006392
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.344277
```

```
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.803619


Table Schema        = NEWTON
Table Name          = SALES
Table Type          = User
Data Partition Id   = 2
Data Object Pages   = 3
Rows Read           = 4
Rows Written        = 1
Overflows           = 0
Page Reorgs         = 0
Table Reorg Information:
  Node number       = 0
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.199971
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.362918
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.821244

Table Reorg Information:
  Node number       = 1
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.223742
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.420741
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.899543

Table Reorg Information:
  Node number       = 2
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
```

```
  Reorg Tablespace  = 3
 Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.179922
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.344277
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.803619
```

例 2:

```
GET SNAPSHOT FOR TABLES ON DPARTDB AT DBPARTITIONNUM 2
```

出力は関係のある表の情報だけを含むように変更されています。

```
            Table Snapshot

First database connect timestamp     = 06/28/2005 13:46:43.617833
Last reset timestamp                 =
Snapshot timestamp                   = 06/28/2005 13:46:51.016787
Database name                        = DPARTDB
Database path                        = /work/sales/NODE0000/SQL00001/
Input database alias                 = DPARTDB
Number of accessed tables            = 3

Table List
 Table Schema        = NEWTON
 Table Name          = SALES
 Table Type          = User
 Data Partition Id   = 0
 Data Object Pages   = 1
 Rows Read           = 0
 Rows Written        = 0
 Overflows           = 0
 Page Reorgs         = 0
 Table Reorg Information:
   Node number       = 2
   Reorg Type        =
        Reclaiming
        Table Reorg
        Allow No Access
        Recluster Via Table Scan
        Reorg Data Only
   Reorg Index       = 0
   Reorg Tablespace  = 3
 Long Temp space ID  = 3
   Start Time        = 06/28/2005 13:46:49.814813
   Reorg Phase       = 3 - Index Recreate
   Max Phase         = 3
   Phase Start Time  = 06/28/2005 13:46:50.344277
   Status            = Completed
   Current Counter   = 0
   Max Counter       = 0
   Completion        = 0
   End Time          = 06/28/2005 13:46:50.803619


 Table Schema        = NEWTON
 Table Name          = SALES
 Table Type          = User
 Data Partition Id   = 1
 Data Object Pages   = 1
 Rows Read           = 0
```

```
Rows Written        = 0
Overflows           = 0
Page Reorgs         = 0
Table Reorg Information:
  Node number       = 2
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.006392
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.344277
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.803619


Table Schema        = NEWTON
Table Name          = SALES
Table Type          = User
Data Partition Id   = 2
Data Object Pages   = 1
Rows Read           = 4
Rows Written        = 1
Overflows           = 0
Page Reorgs         = 0
Table Reorg Information:
  Node number       = 2
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.179922
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.344277
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.803619
```

例 3:

```
SELECT * FROM SYSIBMADM.SNAPLOCK WHERE tabname = 'SALES';
```

出力は、関係のある表の情報のサブセットだけを含むように変更されています。

```
...   TBSP_NAME TABNAME LOCK_OBJECT_TYPE   LOCK_MODE  LOCK_STATUS ...
      --------- ------- ------------------ ---------- -----------
...   PARTTBS   SALES      ROW_LOCK        X          GRNT        ...
...   -         SALES      TABLE_LOCK      IX         GRNT        ...
...   PARTTBS   SALES      TABLE_PART_LOCK IX         GRNT        ...
```

```
...   PARTTBS   SALES      ROW_LOCK        X          GRNT         ...
...   -         SALES      TABLE_LOCK      IX         GRNT         ...
...   PARTTBS   SALES      TABLE_PART_LOCK IX         GRNT         ...
...   PARTTBS   SALES      ROW_LOCK        X          GRNT         ...
...   -         SALES      TABLE_LOCK      IX         GRNT         ...
...   PARTTBS   SALES      TABLE_PART_LOCK IX         GRNT         ...

 9 record(s) selected.
```

この照会の出力 (続き)。

```
... LOCK_ESCALATION LOCK_ATTRIBUTES DATA_PARTITION_ID DBPARTITIONNUM
    --------------- --------------- ----------------- --------------
...               0 INSERT          2                 2
...               0 NONE            -                 2
...               0 NONE            2                 2
...               0 INSERT          0                 0
...               0 NONE            -                 0
...               0 NONE            0                 0
...               0 INSERT          1                 1
...               0 NONE            -                 1
...               0 NONE            1                 1
```

例 4:

```
SELECT * FROM SYSIBMADM.SNAPTAB WHERE tabname = 'SALES';
```

出力は、関係のある表の情報のサブセットだけを含むように変更されています。

```
... TABSCHEMA TABNAME TAB_FILE_ID TAB_TYPE   DATA_OBJECT_PAGES ROWS_WRITTEN ...
... --------- ------- ----------- ---------- ----------------- ------------ ...
... NEWTON    SALES             2 USER_TABLE                 1            1 ...
... NEWTON    SALES             4 USER_TABLE                 1            1 ...
... NEWTON    SALES             3 USER_TABLE                 1            1 ...

 3 record(s) selected.
```

この照会の出力 (続き)。

```
... OVERFLOW_ACCESSES PAGE_REORGS DBPARTITIONNUM TBSP_ID DATA_PARTITION_ID
... ----------------- ----------- -------------- ------- -----------------
...                 0           0              0       3                 0
...                 0           0              2       3                 2
...                 0           0              1       3                 1
```

例 5:

```
SELECT * FROM SYSIBMADM.SNAPTAB_REORG WHERE tabname = 'SALES';;
```

出力は、関係のある表の情報のサブセットだけを含むように変更されています。

```
 REORG_PHASE   REORG_MAX_PHASE REORG_TYPE                                      ...
    ------------- --------------- --------------------------------------------
INDEX_RECREATE               3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
INDEX_RECREATE               3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
INDEX_RECREATE               3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
INDEX_RECREATE               3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
INDEX_RECREATE               3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
INDEX_RECREATE               3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
INDEX_RECREATE               3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
INDEX_RECREATE               3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
INDEX_RECREATE               3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...

 9 record(s) selected.
```

この照会の出力 (続き)。

```
... REORG_STATUS REORG_TBSPC_ID DBPARTITIONNUM DATA_PARTITION_ID
    ------------------------ -------------- ----------------- -----------------
... COMPLETED                3              2                 0
... COMPLETED                3              2                 1
... COMPLETED                3              2                 2
... COMPLETED                3              1                 0
... COMPLETED                3              1                 1
... COMPLETED                3              1                 2
... COMPLETED                3              0                 0
... COMPLETED                3              0                 1
... COMPLETED                3              0                 2
```

例 6: 表の REORG 情報には再編成操作の一部としてエクステントを再利用することに関する情報が含められています。 続く例は関係する出力を示しています。

```
db2 -v "get snapshot for tables on wsdb"

 Table Reorg Information:
   Reorg Type        =
        Reclaim Extents
        Allow Write Access
   Reorg Index       = 0
   Reorg Tablespace  = 0
   Start Time        = 10/22/2008 15:49:35.477532
   Reorg Phase       = 12 - Release
   Max Phase         = 3
```

注: SQLM_DBMON_VERSION9_7 以前のモニターのバージョンからのスナップショット要求はどれも要求クライアントに再利用の REORG 状況を戻しません。

DEADLOCK WITH DETAILS HISTORY イベント・モニターの非アクティブ・ステートメント・トラッキング

すべてのステートメント (およびオプションでデータ値) をトラッキングするデッドロック・イベント・モニターを実行すると、 1 つの作業単位内に非常に多数のステートメントが入っている単一のアプリケーションによって、システムのモニター・ヒープが使い尽くされてしまうことがあります。また、同時に実行するアプリケーションが多数存在する場合も、モニター・ヒープが使い尽くされてしまう可能性があります。

消費されるスペースの量を少なくするため、アプリケーションの非アクティブ・ステートメントの数が特定のしきい値に達すると、非アクティブ・ステートメントはそのアプリケーションによってイベント・モニターに書き出されます。イベント・モニターに書き出された後、それらの非アクティブ・ステートメントによって消費されていたメモリーは解放されます。また、アプリケーションは、システム・モニター・ヒープからメモリーを取得できない場合、その現在のすべての非アクティブ・ステートメントをイベント・モニターに書き出してから、メモリーの取得を再試行します。2 回目の試行に失敗した場合は、メッセージがログに記録され、ステートメント履歴リストの、アプリケーションが処理している作業単位に対応する部分が切り捨てられます。

1 つのアプリケーションが保持する非アクティブ・ステートメントの数のデフォルトの限度は 250 です。このデフォルト値は、レジストリー変数 DB2_MAX_INACT_STMTS を使用して別の値を指定することでオーバーライドでき

ます。この限度に対して異なる値を選択することによって、非アクティブ・ステートメントの情報のために用いられるシステム・モニター・ヒープの量を増減できます。

非アクティブ・ステートメントがイベント・モニターに書き出された場合は、そのことを示すメッセージが db2diag ログ・ファイルに記録されます。非アクティブ・ステートメントの限度を超えた場合は、そのことを示すメッセージが db2diag ログ・ファイルに記録されます。

アプリケーションはデッドロックのコンテキスト以外でも (上記のいずれかのしきい値に達したときに) そのステートメント履歴の項目を記録する可能性があるので、分析のためには、それらの項目をデッドロック発生時に記録されたステートメントのリストと関連付ける手段が必要となります。そのためには、次のようになっているステートメント履歴項目を探すことができます。

- `deadlock_id=` 0
- `participant_no` = 0
- `invocation_id=` デッドロックの呼び出し ID
- `application_id=` デッドロックに関係していたアプリケーションのアプリケーション ID

表イベント・モニターに書き出された場合には、evmon_activates の数値もチェックする必要があります。

注:

- REOPT ALWAYS バインド・オプションを使用してコンパイルされた SQL ステートメントに関しては、デッドロック・イベント情報の中に REOPT コンパイルやステートメント実行のデータ値は含められません。
- コーディネーター・ノードでは、前のセクションで説明した条件に起因して非アクティブ・ステートメントがイベント・モニターに書き出されるとき、書き出されたすべてのレコードのシーケンス値が、処理中の現行の作業単位を反映するものに変更されます。この変更は、このデータを、後からデッドロックによって同じ作業単位内に生成されるデータと整合するために行われます。deadlock_id が 0 であるレコードのシーケンス番号とアプリケーション ID の情報を検索することですべての関連データを収集できるようにするため、こうしたことが行われます。この変更が行われると、シーケンス番号が現行作業単位 ID で上書きされるので、前の作業単位で開始して現行作業でもまだアクティブであるステートメントの作業単位情報は利用できないことになります。この動作はリモート・ノードでは発生しません (すなわち、元の作業単位情報は上書きされません)。したがって、デッドロック・イベント・レコードを、デッドロック前に書き出されたレコードと照合する際は、前の作業単位からのアクティブな WITH HOLD カーソルが関係しているとシーケンス番号が異なる可能性があるので注意が必要です。

Windows Management Instrumentation (WMI) の紹介

管理インフラストラクチャーの規格を制定し、各種のハードウェアおよびソフトウェア管理システムの情報を結合するための方法を提供する、業界イニシアチブが存在します。このイニシアチブは、Web-Based Enterprise Management (WBEM) と呼ばれています。 WBEM は、Desktop Management Task Force (DMTF) 主導の業界標準である Common Information Model (CIM) スキーマを基にしています。

Microsoft® Windows Management Instrumentation (WMI) は、サポートされる Windows プラットフォーム用に WBEM イニシアチブを実現したものです。 WMI は、Windows エンタープライズ・ネットワークで役立ちます。これにより、エンタープライズ・ネットワーク・コンポーネントの保守と管理費用が削減されます。 WMI は以下を提供します。

- Windows の操作、構成、および状況の一貫性のあるモデル。
- 管理情報にアクセスできるようにする COM API。
- 他の Windows 管理サービスと協働する機能。
- 柔軟で拡張可能なアーキテクチャー。これにより、ベンダーは、新しい装置、アプリケーション、その他の機能強化をサポートするための他の WMI プロバイダーを作成できます。
- 情報の詳細な照会を作成するための WMI Query Language (WQL)。
- 管理アプリケーション開発者が Visual Basic または Windows Scripting Host (WSH) スクリプトを作成するための API。

WMI アーキテクチャーには、以下の 2 つの部分があります。

1. 管理インフラストラクチャー。これには、CIM Object Manager (CIMOM) と、CIMOM オブジェクト・リポジトリーと呼ばれる管理データ用の中央ストレージ域が含まれています。 CIMOM を使用すると、アプリケーションは一様な方法で管理データにアクセスできるようになります。
2. WMI プロバイダー。 WMI プロバイダーは、CIMOM と管理下のオブジェクトを仲介するコンポーネントです。 WMI API を使用することにより、WMI プロバイダーは、管理下のオブジェクトのデータを CIMOM に提供し、管理アプリケーションの代わりに要求を処理し、イベント通知を生成します。

Windows Management Instrumentation (WMI) プロバイダーは、管理下のオブジェクトと CIM Object Manager (CIMOM) の仲介機能としての役割を果たす、標準の COM または DCOM サーバーです。 CIMOM が、CIMOM オブジェクト・リポジトリーからは使用できないデータ (またはイベント) に対する要求を管理アプリケーションから受け取ると、 CIMOM はその要求を WMI プロバイダーに転送します。 WMI プロバイダーは、管理対象オブジェクトに対して、それぞれのドメインに固有のデータとイベント通知を提供します。

DB2 データベース・システムと Windows Management Instrumentation の統合

Windows Management Instrumentation (WMI) は、 DB2 パフォーマンス・カウンターを使って、また組み込み PerfMon プロバイダーを使用してスナップショット・モニターにアクセスできます。

WMI は、組み込みレジストリー・プロバイダーを使用して、 DB2 プロファイル・レジストリー変数にアクセスできます。

WMI Software Development Kit (WMI SDK) には、以下の複数の組み込みプロバイダーが含まれています。

- PerfMon プロバイダー
- レジストリー・イベント・プロバイダー
- レジストリー・プロバイダー
- Windows イベント・ログ・プロバイダー
- Win32 プロバイダー
- WDM プロバイダー

WMI は、組み込み Windows イベント・ログ・プロバイダーを使用して、イベント・ログ内の DB2 エラーにアクセスできます。

DB2 データベース・システムには、以下の管理対象オブジェクトにアクセスするための、 DB2 WMI 管理プロバイダーとサンプルの WMI スクリプト・ファイルがあります。

1. 分散インスタンスを含むデータベース・サーバーのインスタンス。以下の操作を実行できます。
   - インスタンスの列挙
   - データベース・マネージャー・パラメーターの構成
   - DB2 サーバー・サービスの開始/停止/状況照会
   - 通信のセットアップまたは確立
2. データベース。以下の操作を実行できます。
   - データベースの列挙
   - データベース・パラメーターの構成
   - データベースの作成/ドロップ
   - データベースのバックアップ/リストア/ロールフォワード

WMI アプリケーションを実行する前に、 DB2 WMI プロバイダーをシステムに登録する必要があります。以下のコマンドを入力して登録を行います。

- mofcomp %DB2PATH%¥bin¥db2wmi.mof

  このコマンドは、DB2 WMI スキーマの定義をシステムにロードします。

- regsvr %DB2PATH%¥bin¥db2wmi.dll

  このコマンドは、DB2 WMI プロバイダー COM DLL を Windows に登録します。

両方のコマンドで、%DB2PATH% は DB2 のインストール先のパスです。また、db2wmi.mof は DB2 WMI スキーマ定義が入っている .MOF ファイルです。

WMI インフラストラクチャーを統合することには、以下のような複数の利点があります。

1. WMI 提供のツールを使用して、 Windows ベースの環境で DB2 サーバーを管理するためのスクリプトを簡単に作成できます。インスタンスのリスト、データベースの作成とドロップ、構成パラメーターの更新などの単純なタスクを実行するための、サンプルの Visual Basic (VBS) スクリプトが提供されています。サンプル・スクリプトは、DB2 Application Development for Windows 製品に組み込まれています。
2. WMI を使用して多くのタスクを実行する強力な管理アプリケーションを作成できます。タスクには以下のものがあります。
   - システム情報の表示
   - DB2 パフォーマンスのモニター
   - DB2 システム・リソース使用量のモニター

   このタイプの管理アプリケーションを使って、システム・イベントと DB2 イベントの両方をモニターすることにより、データベースをよりよく管理できます。
3. 既存の COM および Visual Basic プログラミングの知識とスキルを活用できます。 COM または Visual Basic インターフェースにより、プログラマーは、エンタープライズ管理アプリケーションの開発時間を短縮できます。

Windows パフォーマンス・モニターの紹介

Windows 用の DB2 データベース・マネージャーを使用する場合は、パフォーマンスをモニターする以下のツールを使用できます。

- **DB2 Performance Expert**

  DB2 Performance Expert (マルチプラットフォーム版)、バージョン 1.1 は、 DB2 のデータベース・パフォーマンス関連情報に基づいて自動管理およびソース・チューニングの変更を統合、報告、分析、および推奨します。

- **DB2 ヘルス・センター**

  ヘルス・センターの機能により、パフォーマンス関連情報を処理するためのさまざまな手法が可能になります。これらの機能を、コントロール・センターのパフォーマンス・モニターの一部の機能の代わりに使用することもできます。

  重要: ヘルス・センターは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。 詳しくは、「*DB2 バージョン 9.7 の新機能*」のトピック『コントロール・センター・ツールおよび DB2 管理サーバー (DAS) が推奨されなくなった』を参照してください。

- **Windows Performance Monitor**

  Windows パフォーマンス・モニターを使用すると、データベースとシステム・パフォーマンスの両方をモニターでき、そのシステムに登録されている任意のパフォーマンス・データ提供元から情報を取り出すことができます。さらに Windows は、以下を含めたコンピューター操作のすべてについて、パフォーマンス情報データを提供します。
  - CPU 使用率
  - メモリー使用率
  - ディスクの活動状況
  - ネットワークの活動状況

Windows パフォーマンス・モニターへの DB2 の登録

セットアップ・プログラムは、 DB2 を自動的に Windows パフォーマンス・モニターへ登録します。

Windows パフォーマンス・モニターを使用し、 DB2 データベースおよび DB2® Connect™ のパフォーマンス情報にアクセスできるようにするには、 DB2 (Windows 版) のパフォーマンス・カウンター用の DLL を登録する必要があります。またここで登録しておけば、 Win32 パフォーマンス API を使用してパフォーマンス・データを入手する Windows アプリケーションが他にあれば、そのアプリケーションも使用できるようになります。 DB2 パフォーマンス・カウンターの DLL (DB2Perf.DLL) を、Windows パフォーマンス・モニターにインストールして登録するには、次のように入力します。

```
db2perfi -i
```

DLL を登録するならば、レジストリーのサービス・オプションで、新しいキーを作成することもできます。 1 つの項目には DLL の名前が示され、カウンターをサポートします。他の 3 つの項目には、その DLL に備えられている機能の名前が示されます。それらの機能は、以下のとおりです。

オープン
: 処理中に、DLL がシステムによって初めてロードされるときに呼び出されます。

収集
: DLL からのパフォーマンス情報を要求するときに呼び出されます。

クローズ
: DLL をアンロードするときに呼び出されます。

DB2 パフォーマンス情報へのリモート・アクセスを使用可能にする

ご使用の DB2 (Windows 版) ワークステーションが、別の Windows コンピューターにネットワークで接続されている場合、この節で説明されているフィーチャーを使用できます。

別の DB2 (Windows 版) コンピューターから Windows パフォーマンス・オブジェクトを見るには、 DB2 データベース・マネージャーに管理者のユーザー名とパスワードを登録しなければなりません。(デフォルトの Windows パフォーマンス・モニターのユーザー名である SYSTEM は、 DB2 データベースの予約語なので使用できません。) 名前を登録するには、次のように入力します。

```
db2perfr -r username password
```

注: 使用する username は、 DB2 データベース命名規則に適合しなければなりません。

ユーザー名とパスワードのデータは、レジストリー内のキーに置かれます。このときのセキュリティーは、管理者および SYSTEM アカウントだけがアクセスできるというものです。管理者のパスワードをレジストリーに格納する際のセキュリティー上の問題を防ぐため、データはエンコードされます。

注:

1. いったんユーザー名とパスワードの組み合わせを DB2 データベース・システムに登録すれば、パフォーマンス・モニターのローカル・インスタンスであっても、そのユーザー名とパスワードを使って明示的にログオンします。つまり、DB2 データベース・システムに登録されたユーザー名情報が一致しなければ、パフォーマンス・モニターのローカル・セッションには、 DB2 データベースのパフォーマンス情報が示されないことになります。
2. ユーザー名とパスワードの組み合わせは、 Windows のセキュリティー・データベースに格納されているユーザー名とパスワードと常に一致させる必要があります。Windows セキュリティー・データベース内のユーザー名かパスワードを変更した場合、リモートのパフォーマンス・モニターに使うユーザー名とパスワードの組み合わせを再設定しなければなりません。
3. 登録するには、次のように入力します。

```
db2perfr -u <username> <password>
```

DB2 データベースと DB2 Connect のパフォーマンス値を表示する

パフォーマンス・モニターを使って DB2 データベースおよび DB2 Connect のパフォーマンス値を表示するには、「**追加先**」ボックスから、表示させる値を示すパフォーマンス・カウンターを選択します。このボックスには、パフォーマンス・データを提示するパフォーマンス・オブジェクトのリストが示されます。提供されているカウンターのリストを見るには、特定のオブジェクトを選択してください。

1 つのパフォーマンス・オブジェクトに、複数のインスタンスが存在することもあります。例えば、LogicalDisk オブジェクトには、「% Disk Read Time」や「Disk Bytes/sec」などのカウンターが備えられています。さらに、コンピューター内の論理ドライブ (「C:」や「D:」など) ごとに、1 つのインスタンスがあります。

Windows パフォーマンス・オブジェクト

Windows には、以下のパフォーマンス・オブジェクトがあります。

- **DB2 データベース・マネージャー**

  このオブジェクトは、1 つの Windows インスタンスのための、一般的な情報を提供します。モニターされる DB2 データベース・インスタンスは、オブジェクト・インスタンスとして表されます。

  実用上ならびにパフォーマンス上の理由のため、パフォーマンス情報は、一度に 1 つの DB2 データベース・インスタンスだけから入手されます。パフォーマンス・モニターが示す DB2 データベース・インスタンスは、パフォーマンス・モニターの処理では、 `db2instance` レジストリー変数によって管理されます。同時に複数の DB2 データベース・インスタンスを実行していて、 2 つ以上のパフォーマンス情報を確認する場合、パフォーマンス・モニターのセッションを個別に開始する必要があります。このとき、`db2instance` には、モニターする DB2 データベース・インスタンスごとに対応する値を設定します。

  パーティション・データベース環境を実行している場合は、一度に 1 つのデータベース・パーティション・サーバーからのみ、パフォーマンス情報を入手できます。デフォルトでは、デフォルト・データベース・パーティション (論理ポートが 0 のデータベース・パーティション) のパフォーマンス情報が表示されます。

他のデータベース・パーティションのパフォーマンス情報を表示するには、DB2NODE 環境変数を、モニターするデータベース・パーティションのデータベース・パーティション番号に設定して、パフォーマンス・モニターの他のセッションを開始する必要があります。

- **DB2 データベース**

  このオブジェクトは、特定のデータベースの情報を提供します。現在アクティブなデータベースごとに、情報を利用できます。

- **DB2 アプリケーション**

  このオブジェクトは、特定の DB2 データベース・アプリケーションの情報を提供します。現在アクティブな DB2 データベース・アプリケーションごとに、情報を利用できます。

- **DB2 DCS データベース**

  このオブジェクトは、特定の DCS データベースの情報を提供します。現在アクティブなデータベースごとに、情報を利用できます。

- **DB2 DCS アプリケーション**

  このオブジェクトは、特定の DB2 DCS アプリケーションの情報を提供します。現在アクティブな DB2 DCS アプリケーションごとに、情報を利用できます。

Windows パフォーマンス・モニターによってリストされるオブジェクトは、Windows コンピューターに何がインストールされているか、およびどのアプリケーションがアクティブかによって異なります。例えば、 DB2 データベース・マネージャーをインストールし開始していれば、 DB2 データベース・マネージャー・オブジェクトがリストされます。さらに、そのコンピューターで現在アクティブな DB2 データベースおよびアプリケーションがあれば、 DB2 データベースおよび DB2 アプリケーション・オブジェクトもリストされます。Windows システムを DB2 Connect のゲートウェイとして使っていて、現在アクティブな DCS データベースおよびアプリケーションがいくつかある場合、 DB2 DCS データベースおよび DB2 DCS アプリケーション・オブジェクトがリストされます。

リモートの DB2 データベースのパフォーマンス情報へのアクセス

DB2 パフォーマンス情報にリモートでアクセスできるようにする方法については、すでに説明しました。「**追加先**」ボックスで、モニターする別のコンピューターを選択してください。これにより、そのコンピューター上で使用できるすべてのパフォーマンス・オブジェクトのリストが表示されます。

リモート・コンピューターで DB2 パフォーマンス・オブジェクトをモニターできるようにするには、そのコンピューターにインストールされている DB2 データベースまたは DB2 Connect コードのレベルが、バージョン 6 以上でなければなりません。

DB2 パフォーマンス値をリセットする

アプリケーションで DB2 モニター API を呼び出すと、戻される情報は、通常は DB2 データベース・サーバー開始以降の累積値になります。これは、以下のような場合に役立ちます。

- パフォーマンス値をリセットする
- テストを実行する
- その値をもう一度リセットする
- テストを再実行する

データベースのパフォーマンス値をリセットするには、 db2perfc プログラムを使用します。次のように入力してください。

```
db2perfc
```

デフォルトでは、これによりアクティブな DB2 データベースすべてのパフォーマンス値がリセットされます。ただし、リセットするデータベースのリストを指定することも可能です。また -d オプションを使用して、 DCS データベースのパフォーマンス値をリセットするよう指定することもできます。以下に例を示します。

```
db2perfc
db2perfc dbalias1 dbalias2 ... dbaliasn

db2perfc  -d
db2perfc  -d  dbalias1 dbalias2 ... dbaliasn
```

最初の例では、すべてのアクティブな DB2 データベースのパフォーマンス値がリセットされます。次の例では、特定の DB2 データベースの値がリセットされます。3 番目の例では、すべてのアクティブな DB2 DCS データベースのパフォーマンス値がリセットされます。最後の例では、特定の DB2 DCS データベースの値がリセットされます。

db2perfc プログラムは、関連する DB2 データベース・サーバー・インスタンス (つまり、 db2perfc 実行時のセッションの DB2INSTANCE に保持されるインスタンス) に関するデータベース・パフォーマンス情報に現在アクセスしているすべてのプログラムの値をリセットします。

db2perfc を呼び出すと、 db2perfc コマンドの実行中に DB2 データベースのパフォーマンス情報にリモートでアクセスしているユーザーがいる場合、そのユーザーに表示される値もリセットされます。

注: sqlmrset という DB2 データベース API を使用すれば、アプリケーションでグローバルではなくローカルに表示される特定のデータベースの値を、アプリケーション側でリセットできます。

未確定トランザクション・マネージャーの概要

「未確定トランザクション・マネージャー」ウィンドウを使用して、未確定トランザクションを処理します。ウィンドウには、選択されたデータベースおよび選択された 1 つ以上のパーティションのすべての未確定トランザクションがリストされます。

重要: 未確定トランザクション・マネージャーは、バージョン 9.7 で非推奨となり、将来のリリースで除去される可能性があります。 詳しくは、「*DB2 バージョン 9.7 の新機能*」のトピック『コントロール・センター・ツールおよび DB2 管理サーバー (DAS) が推奨されなくなった』を参照してください。

未確定トランザクションとは、未確定状態のままにされたグローバル・トランザクションのことです。 DB2 は、リソース所有者 (データベース管理者など) がトランザクション・マネージャーによる再同期アクションの実行を待てない場合に、データベース管理者が未確定トランザクションに対して実行できるヒューリスティック・アクションを提供します。例えば通信回線が切断され、必要なリソース (例えばトランザクションが使用している表および索引に対するロック、ログ・スペース、ストレージなど) が未確定トランザクションと結びついてしまっている場合にこの状態が起こり得ます。

本来トランザクション・マネージャーが再同期アクションを開始することが望ましいのですが、ユーザー自身が未確定トランザクションに対してヒューリスティック・アクションを実行しなければならない場合もあるかもしれません。実行する場合には十分な注意を要します。このヒューリスティック・アクションは最後の手段として使用します。使用の際は、次のガイドラインに従ってください。

- トランザクション ID の *gtrid* の部分はグローバル・トランザクション ID で、これはグローバル・トランザクションに関係する他のリソース・マネージャー (RM) のものと同じです。
- アプリケーションおよびオペレーティング環境の知識を活用し、関係する他のリソース・マネージャーを識別します。
- トランザクション・マネージャーが CICS® で、リソース・マネージャーが CICS リソースだけである場合、ヒューリスティック・ロールバックを実行します。
- トランザクション・マネージャーが CICS でない場合、それを使用して、未確定トランザクションと同じ *gtrid* を持つトランザクションの状況を判別します。
- 少なくとも 1 つのリソース・マネージャーがコミットまたはロールバックを行った場合、ヒューリスティック・コミットまたはロールバックを実行します。
- すべてのトランザクションが Prepared 状態にある場合、ヒューリスティック・ロールバックを実行します。
- 少なくともリソース・マネージャーの 1 つが利用不可になっている場合、ヒューリスティック・ロールバックを実行します。

Intel プラットフォーム上で未確定トランザクション・マネージャーを開くには、**「スタート」**メニューから、**「スタート」->「プログラム」->「IBM DB2」->「モニター・ツール (Monitoring Tools)」->「未確定トランザクション・マネージャー」**とクリックします。

UNIX または Intel のコマンド行を使って未確定トランザクション・マネージャーを開くには、次のコマンドを実行します。

```
db2indbt
```

未確定トランザクションに対しては、次のヒューリスティック・アクションを実行できます。

- 取り消し

  これは、ログ・レコードを除去し、ログ・ページを解放することによって、ヒューリスティックに完了したトランザクションの知識をリソース・マネージャーが消去できるようにします。ヒューリスティックに完了したトランザクションとは、ヒューリスティックにコミットまたはロールバックされたトランザクションのことです。取り消しアクションは、選択されたデータベースおよび選択された

1 つ以上のパーティションについての、ヒューリスティックにコミットまたはロールバックされるトランザクションに対して使用できます。未確定トランザクションを取り消すには、データベースおよびパーティションを選択した後、「**コミット**」または「**ロールバック**」状態になっているトランザクションを右クリックして、ポップアップ・メニューから「**取り消し**」を選択します。確認メッセージが表示されます。

- コミット

  これは、コミット準備が整っている未確定トランザクションをコミットします。操作が成功すると、トランザクションの状態が「ヒューリスティックにコミット」になります。未確定トランザクションをコミットするには、データベースおよびパーティションを選択した後、「**未確定**」または「**コミット確認通知の欠落**」状態になっているトランザクションを右クリックして、ポップアップ・メニューから「**コミット**」を選択します。確認メッセージが表示されます。

- ロールバック

  これは、準備済みの未確定トランザクションをロールバックします。操作が成功すると、トランザクションの状態が「ヒューリスティックにロールバック」になります。未確定トランザクションをロールバックするには、データベースおよびパーティションを選択した後、「**未確定**」または「**終了**」状態になっているトランザクションを右クリックして、ポップアップ・メニューから「**ロールバック**」を選択します。確認メッセージが表示されます。

未確定トランザクションに対してこれらのアクションを実行するには、SYSADM または DBADM 権限が必要です。

「未確定トランザクション・マネージャー」ウィンドウの列には、さまざまな方法で未確定トランザクションを編成し、表示するための名前付きビューがあります。次に、このインターフェース内の各列についてリストします。

状況　トランザクションの未確定の状況。つまり、コミット (c)、終了 (e)、未確定 (i)、コミット確認通知の欠落 (m)、ロールバック (r) を示します。

コミット
この状態にあるトランザクションは、ヒューリスティックにコミット済みであることを意味します。

終了　この状態にあるトランザクションは、タイムアウトになった可能性があることを意味します。

未確定　この状態にあるトランザクションは、コミットまたはロールバックされるのを待機していることを意味します。

コミット確認通知の欠落
トランザクションをコミットする前に確認通知を受け取るのをトランザクション・マネージャーが待機していることを意味します。

ロールバック
この状態にあるトランザクションは、ヒューリスティックなロールバックが完了していることを意味します。

タイム・スタンプ
トランザクションが準備済み (未確定) の状態に入ったときのサーバー上のタイム・スタンプ。時刻はクライアントのローカル時刻で表されます。

トランザクション ID
グローバル・トランザクションを一意的に識別するためにトランザクション・マネージャーが割り当てる XA ID。

アプリケーション ID
データベース・マネージャーがこのトランザクションに割り当てるアプリケーション ID。

許可 ID
トランザクションを実行したユーザーのユーザー ID。

シーケンス番号
アプリケーション ID の拡張としてデータベース・マネージャーが割り当てるシーケンス番号。

パーティション
未確定トランザクションが存在するパーティション。

発信元 パーティション・データベース環境で、トランザクションの発信元が XA であるかまたは DB2 であるかを示します。

ログ・フル
ログ・フル状態の原因がこのトランザクションであるかどうかを示します。

タイプ 各未確定トランザクションにおけるデータベースの役割を示すタイプ情報。

- **TM** は、未確定トランザクションがデータベースをトランザクション・マネージャーのデータベースとして使用していることを示しています。
- **RM** は、未確定トランザクションがデータベースをリソース・マネージャーとして使用していることを示しています。これは、トランザクションに関係しているデータベースの 1 つであるものの、トランザクション・マネージャーのデータベースではないことを意味しています。

第 2 部 モニター・エレメント

第 7 章 モニター表関数で報告されるモニター・エレメント

DB2 バージョン 9.7 では、新しいモニター表関数を介して報告されるいくつかのモニター・エレメントが導入されました。

これらのモニター・エレメントは、システム処理、アクティビティー、および (表、表スペース、表スペース・コンテナー、バッファー・プールなどの) データ・オブジェクトについての情報を提供します。

- 389 ページの『audit_events_total - 監査イベントの合計 : モニター・エレメント』
- 390 ページの『audit_file_write_wait_time - 監査ファイル書き込み待機時間 : モニター・エレメント』
- 391 ページの『audit_file_writes_total - 書き込まれた監査ファイルの合計 : モニター・エレメント』
- 392 ページの『audit_subsystem_wait_time - 監査サブシステム待機時間 : モニター・エレメント』
- 394 ページの『audit_subsystem_waits_total - 監査サブシステム待機の合計 : モニター・エレメント』
- 397 ページの『auto_storage_hybrid - ハイブリッド自動ストレージの表スペース標識 : モニター・エレメント』
- 398 ページの『automatic - バッファー・プール自動 : モニター・エレメント』
- 399 ページの『block_ios - ブロック入出力要求数 : モニター・エレメント』
- 401 ページの『boundary_leaf_node_splits - 境界リーフ・ノードの分割 : モニター・エレメント』
- 402 ページの『bp_name - バッファー・プール名 : モニター・エレメント』
- 409 ページの『client_acctng - クライアント・アカウンティング・ストリング : モニター・エレメント』
- 409 ページの『client_applname - クライアント・アプリケーション名 : モニター・エレメント』
- 411 ページの『client_idle_wait_time - クライアントのアイドル待機時間 : モニター・エレメント』
- 414 ページの『client_userid - クライアントのユーザー ID : モニター・エレメント』
- 415 ページの『client_wrkstnname - クライアント・ワークステーション名 : モニター・エレメント』
- 418 ページの『comp_env_desc コンパイル環境 : モニター・エレメント』
- 421 ページの『concurrentdbcoordactivities_db_threshold_id 並行データベース・コーディネーター・アクティビティーのデータベースしきい値 ID : モニター・エレメント』
- 422 ページの『concurrentdbcoordactivities_db_threshold_queued 並行データベース・コーディネーター・アクティビティーのデータベースしきい値によるキュー待機 : モニター・エレメント』
- 422 ページの『concurrentdbcoordactivities_db_threshold_value 並行データベース・コーディネーター・アクティビティーのデータベースしきい値 : モニター・エレメント』
- 422 ページの『concurrentdbcoordactivities_db_threshold_violated 並行データベース・コーディネーター・アクティビティーのデータベースしきい値の違反 : モニター・エレメント』
- 423 ページの『concurrentdbcoordactivities_subclass_threshold_id 並行データベース・コーディネーター・アクティビティーのサービス・サブクラスしきい値 ID : モニター・エレメント』

- 423 ページの『concurrentdbcoordactivities_subclass_ threshold_queued 並行データベース・コーディネーター・アクティビティーのサービス・サブクラスしきい値によるキュー待機 : モニター・エレメント』
- 424 ページの『concurrentdbcoordactivities_subclass_ threshold_value 並行データベース・コーディネーター・アクティビティーのサービス・サブクラスしきい値 : モニター・エレメント』
- 424 ページの『concurrentdbcoordactivities_subclass_ threshold_violated 並行データベース・コーディネーター・アクティビティーのサービス・サブクラスしきい値の違反 : モニター・エレメント』
- 425 ページの『concurrentdbcoordactivities_superclass_ threshold_id 並行データベース・コーディネーター・アクティビティーのサービス・スーパークラスしきい値 ID : モニター・エレメント』
- 425 ページの『concurrentdbcoordactivities_superclass_ threshold_queued 並行データベース・コーディネーター・アクティビティーのサービス・スーパークラスしきい値によるキュー待機 : モニター・エレメント』
- 425 ページの『concurrentdbcoordactivities_superclass_ threshold_value 並行データベース・コーディネーター・アクティビティーのサービス・スーパークラスしきい値 : モニター・エレメント』
- 426 ページの『concurrentdbcoordactivities_superclass_ threshold_violated 並行データベース・コーディネーター・アクティビティーのサービス・スーパークラスしきい値の違反 : モニター・エレメント』
- 426 ページの『concurrentdbcoordactivities_work_action_set_ threshold_id 並行データベース・コーディネーター・アクティビティーの作業アクション・セットしきい値 ID : モニター・エレメント』
- 427 ページの『concurrentdbcoordactivities_work_action_set_ threshold_queued 並行データベース・コーディネーター・アクティビティーの作業アクション・セットしきい値によるキュー待機 : モニター・エレメント』
- 427 ページの『concurrentdbcoordactivities_work_action_set_ threshold_value 並行データベース・コーディネーター・アクティビティーの作業アクション・セットしきい値 : モニター・エレメント』
- 428 ページの『concurrentdbcoordactivities_work_action_set_ threshold_violated 並行データベース・コーディネーター・アクティビティーの作業アクション・セットしきい値の違反 : モニター・エレメント』
- 430 ページの『container_accessible - コンテナーのアクセス可能性 : モニター・エレメント』
- 431 ページの『container_id - コンテナー ID : モニター・エレメント』
- 431 ページの『container_name - コンテナー名 : モニター・エレメント』
- 431 ページの『container_stripe_set - コンテナー・ストライプ・セット : モニター・エレメント』
- 432 ページの『container_total_pages - コンテナー内の合計ページ数 : モニター・エレメント』
- 432 ページの『container_type - コンテナー・タイプ : モニター・エレメント』
- 433 ページの『container_usable_pages - コンテナー内の使用可能なページ数 : モニター・エレメント』

- 438 ページの『coord_member - コーディネーター・メンバー : モニター・エレメント』
- 443 ページの『cputime_threshold_id - CPU 時間しきい値 ID : モニター・エレメント』
- 444 ページの『cputime_threshold_value - CPU 時間しきい値 : モニター・エレメント』
- 444 ページの『cputime_threshold_violated - CPU 時間しきい値の違反 : モニター・エレメント』
- 444 ページの『cputimeinsc_threshold_id - サービス・クラス内の CPU 時間しきい値 ID : モニター・エレメント』
- 445 ページの『cputimeinsc_threshold_value - サービス・クラス内の CPU 時間しきい値 : モニター・エレメント』
- 445 ページの『cputimeinsc_threshold_violated - サービス・クラス内の CPU 時間しきい値の違反 : モニター・エレメント』
- 448 ページの『current_extent - 現在移動中のエクステント : モニター・エレメント』
- 449 ページの『data_partition_id - データ・パーティション ID : モニター・エレメント』
- 455 ページの『db_storage_path_state - ストレージ・パスの状態 : モニター・エレメント』
- 455 ページの『db_storage_path_with_dpe - データベース・パーティション式を含むストレージ・パス : モニター・エレメント』
- 456 ページの『db_work_action_set_id データベース作業アクション・セット ID : モニター・エレメント』
- 456 ページの『db_work_class_id データベース作業クラス ID : モニター・エレメント』
- 459 ページの『deadlocks - デッドロック検出数 : モニター・エレメント』
- 461 ページの『del_keys_cleaned - クリーンアップされた疑似削除キー : モニター・エレメント』
- 463 ページの『diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間 : モニター・エレメント』
- 464 ページの『diaglog_writes_total - 診断ログ・ファイル書き込みの合計 : モニター・エレメント』
- 465 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』
- 467 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』
- 468 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』
- 470 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』
- 471 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』
- 473 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』
- 477 ページの『eff_stmt_text - 有効なステートメント・テキスト : モニター・エレメント』

- 477 ページの『effective_isolation - 有効な分離 : モニター・エレメント』
- 478 ページの『effective_lock_timeout - 有効なロック・タイムアウト : モニター・エレメント』
- 478 ページの『effective_query_degree - 有効な照会の度合い : モニター・エレメント』
- 479 ページの『empty_pages_deleted - 削除された空ページ : モニター・エレメント』
- 479 ページの『empty_pages_reused - 再利用された空ページ : モニター・エレメント』
- 480 ページの『entry_time - エントリー時間 : モニター・エレメント』
- 480 ページの『estimatedsqlcost_threshold_id - 見積もり SQL コストしきい値 ID : モニター・エレメント』
- 480 ページの『estimatedsqlcost_threshold_value - 見積もり SQL コストしきい値 : モニター・エレメント』
- 481 ページの『estimatedsqlcost_threshold_violated - 見積もり SQL コストしきい値の違反 : モニター・エレメント』
- 483 ページの『executable_id 実行可能 ID : モニター・エレメント』
- 485 ページの『fcm_message_recv_volume - FCM メッセージ受信ボリューム : モニター・エレメント』
- 486 ページの『fcm_message_recv_wait_time - FCM メッセージの受信待機時間 : モニター・エレメント』
- 487 ページの『fcm_message_recvs_total - FCM メッセージ受信の合計 : モニター・エレメント』
- 487 ページの『fcm_message_send_volume - FCM メッセージ送信ボリューム : モニター・エレメント』
- 488 ページの『fcm_message_send_wait_time - FCM メッセージの送信待機時間 : モニター・エレメント』
- 489 ページの『fcm_message_sends_total - FCM メッセージ送信の合計 : モニター・エレメント』
- 490 ページの『fcm_recv_volume - FCM 受信ボリューム : モニター・エレメント』
- 491 ページの『fcm_recv_wait_time - FCM 受信待機時間 : モニター・エレメント』
- 492 ページの『fcm_recvs_total - FCM 受信の合計 : モニター・エレメント』
- 493 ページの『fcm_send_volume - FCM 送信ボリューム : モニター・エレメント』
- 494 ページの『fcm_send_wait_time - FCM 送信待機時間 : モニター・エレメント』
- 495 ページの『fcm_sends_total - FCM 送信の合計 : モニター・エレメント』
- 496 ページの『fcm_tq_recv_volume - FCM 表キュー受信ボリューム : モニター・エレメント』
- 497 ページの『fcm_tq_recv_wait_time - FCM 表キュー受信待機時間 : モニター・エレメント』

- 498 ページの『fcm_tq_recvs_total - FCM 表キュー受信の合計 : モニター・エレメント』
- 499 ページの『fcm_tq_send_volume - FCM 表キュー送信ボリューム : モニター・エレメント』
- 500 ページの『fcm_tq_send_wait_time - FCM 表キュー送信待機時間 : モニター・エレメント』
- 501 ページの『fcm_tq_sends_total - FCM 表キュー送信の合計 : モニター・エレメント』
- 502 ページの『files_closed - クローズしたデータベース・ファイル : モニター・エレメント』
- 504 ページの『fs_caching - ファイル・システム・キャッシング : モニター・エレメント』
- 505 ページの『fs_id - 固有のファイル・システム識別番号 : モニター・エレメント』
- 505 ページの『fs_total_size - ファイル・システムの合計サイズ : モニター・エレメント』
- 506 ページの『fs_used_size - ファイル・システム上で使用されるスペースの量 : モニター・エレメント』
- 525 ページの『iid - 索引 ID : モニター・エレメント』
- 526 ページの『include_col_updates - 列の更新の組み込み : モニター・エレメント』
- 527 ページの『index_only_scans - 索引のみのスキャン : モニター・エレメント』
- 527 ページの『index_scans - 索引スキャン : モニター・エレメント』
- 527 ページの『index_tbsp_id - 索引表スペース ID : モニター・エレメント』
- 529 ページの『insert_timestamp - ステートメント挿入タイムスタンプ : モニター・エレメント』
- 532 ページの『int_node_splits - 中間ノードの分割 : モニター・エレメント』
- 535 ページの『invocation_id - 呼び出し ID : モニター・エレメント』
- 536 ページの『ipc_recv_volume - プロセス間通信の受信ボリューム : モニター・エレメント』
- 537 ページの『ipc_recv_wait_time - プロセス間通信受信待機時間 : モニター・エレメント』
- 537 ページの『ipc_recvs_total - プロセス間通信受信の合計 : モニター・エレメント』
- 538 ページの『ipc_send_volume - プロセス間通信の送信ボリューム : モニター・エレメント』
- 539 ページの『ipc_send_wait_time - プロセス間通信送信待機時間 : モニター・エレメント』
- 540 ページの『ipc_sends_total - プロセス間通信送信の合計 : モニター・エレメント』
- 541 ページの『key_updates - キーの更新 : モニター・エレメント』

- 542 ページの『last_extent - 移動した最終エクステント : モニター・エレメント』
- 543 ページの『last_reference_time - 最終参照時刻 : モニター・エレメント』
- 546 ページの『local_start_time - ローカル開始時刻 : モニター・エレメント』
- 549 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』
- 558 ページの『lock_timeouts - ロック・タイムアウト数 : モニター・エレメント』
- 560 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』
- 562 ページの『lock_waits - ロック待機数 : モニター・エレメント』
- 565 ページの『log_buffer_wait_time - ログ・バッファー待機時間 : モニター・エレメント』
- 566 ページの『log_disk_wait_time - ログ・ディスク待機時間 : モニター・エレメント』
- 567 ページの『log_disk_waits_total - ログ・ディスク待機の合計 : モニター・エレメント』
- 572 ページの『long_tbsp_id - LONG 表スペース ID : モニター・エレメント』
- 586 ページの『member - データベース・メンバー・モニター・エレメント』
- 588 ページの『nesting_level - ネスト・レベル : モニター・エレメント』
- 589 ページの『nleaf - リーフ・ページ数 : モニター・エレメント』
- 590 ページの『nlevels - 索引レベル数 : モニター・エレメント』
- 591 ページの『nonboundary_leaf_node_splits - 非境界リーフ・ノードの分割 : モニター・エレメント』
- 592 ページの『num_executions - ステートメント実行回数 : モニター・エレメント』
- 593 ページの『num_exec_with_metrics メトリックが収集された実行数 : モニター・エレメント』
- 593 ページの『num_extents_left プロセス残エクステント数 : モニター・エレメント』
- 593 ページの『num_extents_moved - 移動したエクステントの数 : モニター・エレメント』
- 594 ページの『num_log_buffer_full - フル・ログ・バッファーの回数 : モニター・エレメント』
- 598 ページの『num_remaps 再マップ数 : モニター・エレメント』
- 607 ページの『overflow_accesses - オーバーフロー・レコードへのアクセス : モニター・エレメント』
- 608 ページの『overflow_creates - オーバーフローの作成 : モニター・エレメント』
- 608 ページの『package_name - パッケージ名 : モニター・エレメント』
- 609 ページの『package_schema - パッケージ・スキーマ : モニター・エレメント』

- 609 ページの『package_version_id - パッケージ・バージョン : モニター・エレメント』
- 610 ページの『page_allocations - ページ割り振り : モニター・エレメント』
- 611 ページの『pages_from_block_ios - ブロック入出力によって読み取られたページ数の合計 : モニター・エレメント』
- 611 ページの『pages_from_vectored_ios - ベクトル化入出力によって読み取られたページ数の合計 : モニター・エレメント』
- 612 ページの『pages_merged - マージされたページ : モニター・エレメント』
- 612 ページの『pages_read - 読み取られたページの数 : モニター・エレメント』
- 612 ページの『pages_written - 書き込まれたページの数 : モニター・エレメント』
- 613 ページの『parent_activity_id 親アクティビティー ID : モニター・エレメント』
- 613 ページの『parent_uow_id 親作業単位 ID : モニター・エレメント』
- 621 ページの『pool_async_data_read_reqs - バッファー・プール非同期読み取り要求 : モニター・エレメント』
- 622 ページの『pool_async_data_reads バッファー・プール非同期データ読み取り : モニター・エレメント』
- 623 ページの『pool_async_data_writes - バッファー・プール非同期データ書き込み : モニター・エレメント』
- 624 ページの『pool_async_index_read_reqs - バッファー・プール非同期索引読み取り要求 : モニター・エレメント』
- 624 ページの『pool_async_index_reads - バッファー・プール非同期索引読み取り : モニター・エレメント』
- 625 ページの『pool_async_index_writes - バッファー・プール非同期索引書き込み : モニター・エレメント』
- 628 ページの『pool_async_xda_read_reqs - バッファー・プール非同期 XDA 読み取り要求 : モニター・エレメント』
- 629 ページの『pool_async_xda_reads - バッファー・プール非同期 XDA データ読み取り : モニター・エレメント』
- 630 ページの『pool_async_xda_writes - バッファー・プール非同期 XDA データ書き込み : モニター・エレメント』
- 632 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』
- 634 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』
- 635 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』
- 638 ページの『pool_drty_pg_steal_clns - 起動されたバッファー・プール・ビクティム・ページ・クリーナー : モニター・エレメント』
- 639 ページの『pool_drty_pg_thrsh_clns - 起動されたバッファー・プールしきい値クリーナー : モニター・エレメント』

- 641 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』
- 643 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』
- 645 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』
- 647 ページの『pool_lsn_gap_clns - 起動されたバッファー・プール・ログ・スペース・クリーナー : モニター・エレメント』
- 648 ページの『pool_no_victim_buffer - バッファー・プールの非ビクティム・バッファー数 : モニター・エレメント』
- 649 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』
- 652 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』
- 654 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』
- 655 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』
- 657 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』
- 659 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』
- 661 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』
- 663 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』
- 665 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』
- 667 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』
- 669 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』
- 671 ページの『post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント』
- 674 ページの『post_threshold_sorts - ポストしきい値ソート : モニター・エレメント』
- 676 ページの『prep_time 準備時間 : モニター・エレメント』
- 683 ページの『pseudo_deletes - 疑似削除 : モニター・エレメント』
- 684 ページの『pseudo_empty_pages - 疑似空ページ : モニター・エレメント』
- 684 ページの『qp_query_id - Query patroller 照会 ID : モニター・エレメント』
- 685 ページの『query_cost_estimate - 照会コストの見積もり : モニター・エレメント』

- 691 ページの『reclaimable_space_enabled - 再利用可能なスペースが有効な標識 : モニター・エレメント』
- 704 ページの『root_node_splits - ルート・ノードの分割 : モニター・エレメント』
- 704 ページの『routine_id - ルーチン ID : モニター・エレメント』
- 704 ページの『rows_deleted - 削除行数 : モニター・エレメント』
- 705 ページの『rows_inserted - 挿入行数 : モニター・エレメント』
- 706 ページの『rows_modified 変更行数 : モニター・エレメント』
- 707 ページの『rows_read - 読み取り行数 : モニター・エレメント』
- 709 ページの『rows_returned 戻り行数 : モニター・エレメント』
- 712 ページの『rows_updated - 更新行数 : モニター・エレメント』
- 713 ページの『rqsts_completed_total - 完了した要求の合計 : モニター・エレメント』
- 714 ページの『sc_work_action_set_id サービス・クラス作業アクション・セット ID : モニター・エレメント』
- 715 ページの『sc_work_class_id サービス・クラス作業クラス ID : モニター・エレメント』
- 716 ページの『section_number - セクション番号 : モニター・エレメント』
- 718 ページの『section_type - セクション・タイプ標識 : モニター・エレメント』
- 723 ページの『service_class_id サービス・クラス ID : モニター・エレメント』
- 724 ページの『service_subclass_name サービス・サブクラス名 : モニター・エレメント』
- 725 ページの『service_superclass_name サービス・スーパークラス名 : モニター・エレメント』
- 731 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』
- 737 ページの『sqlrowsread_threshold_id - SQL 読み取り行数しきい値 ID : モニター・エレメント』
- 737 ページの『sqlrowsread_threshold_value - SQL 読み取り行数しきい値 : モニター・エレメント』
- 737 ページの『sqlrowsread_threshold_violated - SQL 読み取り行数しきい値の違反 : モニター・エレメント』
- 738 ページの『sqlrowsreadinsc_threshold_id - サービス・クラス内の SQL 読み取り行数しきい値 ID : モニター・エレメント』
- 738 ページの『sqlrowsreadinsc_threshold_value - サービス・クラス内の SQL 読み取り行数しきい値 : モニター・エレメント』
- 738 ページの『sqlrowsreadinsc_threshold_violated - サービス・クラス内の SQL 読み取り行数しきい値の違反 : モニター・エレメント』
- 739 ページの『sqlrowsreturned_threshold_id - 戻される SQL 読み取り行数しきい値 ID : モニター・エレメント』
- 739 ページの『sqlrowsreturned_threshold_value - 戻される SQL 読み取り行数しきい値 : モニター・エレメント』

- 740 ページの『sqlrowsreturned_threshold_violated - 戻される SQL 読み取り行数しきい値の違反 : モニター・エレメント』
- 740 ページの『sqltempspace_threshold_id - SQL 一時スペースしきい値 ID : モニター・エレメント』
- 740 ページの『sqltempspace_threshold_value - SQL 一時スペースしきい値 : モニター・エレメント』
- 741 ページの『sqltempspace_threshold_violated - SQL 一時スペースしきい値の違反 : モニター・エレメント』
- 753 ページの『stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID』
- 757 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』
- 765 ページの『tab_file_id - 表ファイル ID : モニター・エレメント』
- 766 ページの『tab_type - 表タイプ : モニター・エレメント』
- 766 ページの『table_file_id - 表ファイル ID : モニター・エレメント』
- 766 ページの『table_name - 表名 : モニター・エレメント』
- 768 ページの『table_scans - 表スキャン : モニター・エレメント』
- 768 ページの『table_schema - 表スキーマ名 : モニター・エレメント』
- 769 ページの『table_type - 表タイプ : モニター・エレメント』
- 770 ページの『tablespace_auto_resize_enabled - 表スペースの自動サイズ変更可能 : モニター・エレメント』
- 771 ページの『tablespace_content_type - 表スペースのコンテンツ・タイプ : モニター・エレメント』
- 771 ページの『tablespace_cur_pool_id - 現在使用中のバッファー・プール : モニター・エレメント』
- 772 ページの『tablespace_extent_size - 表スペースのエクステント・サイズ : モニター・エレメント』
- 772 ページの『tablespace_free_pages 表スペース内のフリー・ページ数 : モニター・エレメント』
- 773 ページの『tablespace_id - 表スペース ID : モニター・エレメント』
- 776 ページの『tablespace_name - 表スペース名 : モニター・エレメント』
- 777 ページの『tablespace_next_pool_id - 次の始動時に使用されるバッファー・プール : モニター・エレメント』
- 778 ページの『tablespace_page_size - 表スペースのページ・サイズ : モニター・エレメント』
- 779 ページの『tablespace_page_top 表スペース最高水準点 : モニター・エレメント』
- 779 ページの『tablespace_paths_dropped - ドロップされたパスを使用している表スペース : モニター・エレメント』
- 780 ページの『tablespace_pending_free_pages 表スペース内のペンディング・フリー・ページ数 : モニター・エレメント』
- 780 ページの『tablespace_prefetch_size - 表スペースのプリフェッチ・サイズ : モニター・エレメント』

- 782 ページの『tablespace_rebalancer_mode - リバランサー・モード : モニター・エレメント』
- 784 ページの『tablespace_state 表スペースの状態 : モニター・エレメント』
- 786 ページの『tablespace_total_pages 表スペース内の合計ページ数 : モニター・エレメント』
- 787 ページの『tablespace_type - 表スペース・タイプ : モニター・エレメント』
- 787 ページの『tablespace_usable_pages 表スペース内の使用可能ページ数 : モニター・エレメント』
- 788 ページの『tablespace_used_pages 表スペース内の使用されているページ数 : モニター・エレメント』
- 788 ページの『tablespace_using_auto_storage - 自動ストレージが使用可能な表スペース : モニター・エレメント』
- 789 ページの『tbsp_max_page_top - 最大表スペース・ページの最高水準点 : モニター・エレメント』
- 789 ページの『tcpip_recv_volume - TCP/IP 受信ボリューム : モニター・エレメント』
- 790 ページの『tcpip_recv_wait_time - TCP/IP 受信待機時間 : モニター・エレメント』
- 791 ページの『tcpip_recvs_total - TCP/IP 受信の合計 : モニター・エレメント』
- 792 ページの『tcpip_send_volume - TCP/IP 送信ボリューム : モニター・エレメント』
- 793 ページの『tcpip_send_wait_time - TCP/IP 送信待機時間 : モニター・エレメント』
- 793 ページの『tcpip_sends_total - TCP/IP 送信の合計 : モニター・エレメント』
- 801 ページの『total_act_time - 合計アクティビティー時間 : モニター・エレメント』
- 801 ページの『total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント』
- 802 ページの『total_app_rqst_time - 合計アプリケーション要求時間 : モニター・エレメント』
- 804 ページの『total_cpu_time - 合計 CPU 時間 : モニター・エレメント』
- 805 ページの『total_move_time 合計エクステント移動時間 : モニター・エレメント』
- 808 ページの『total_rqst_mapped_in - マッピングの際の要求の合計 : モニター・エレメント』
- 808 ページの『total_rqst_mapped_out - マッピングの際に除外された要求の合計 : モニター・エレメント』
- 809 ページの『total_rqst_time - 合計要求時間 : モニター・エレメント』
- 810 ページの『total_section_sorts - セクションのソートの合計: モニター・エレメント』
- 811 ページの『total_section_sort_proc_time - セクションのソート処理時間の合計 : モニター・エレメント』

- 812 ページの『total_section_sort_time - セクションのソート時間の合計 : モニター・エレメント』
- 814 ページの『total_sorts - ソート合計 : モニター・エレメント』
- 818 ページの『total_wait_time - 合計待機時間 : モニター・エレメント』
- 824 ページの『tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数 : モニター・エレメント』
- 826 ページの『unread_prefetch_pages - 読み取り不能プリフェッチ・ページ : モニター・エレメント』
- 828 ページの『uow_id 作業単位 ID : モニター・エレメント』
- 833 ページの『utility_id ユーティリティー ID』
- 835 ページの『valid セクション妥当性インディケーター : モニター・エレメント』
- 836 ページの『vectored_ios - ベクトル化入出力要求数 : モニター・エレメント』
- 837 ページの『wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て : モニター・エレメント』
- 837 ページの『wlm_queue_time_total - ワークロード・マネージャー合計キュー時間 : モニター・エレメント』
- 840 ページの『workload_id ワークロード ID : モニター・エレメント』
- 841 ページの『workload_name ワークロード名 : モニター・エレメント』
- 842 ページの『workload_occurrence_id ワークロード・オカレンス ID : モニター・エレメント』
- 842 ページの『workload_occurrence_state - ワークロード・オカレンスの状態 : モニター・エレメント』

第 8 章 要求モニター・エレメント

データベース・システム (特に、アプリケーション要求を処理するためにデータ・サーバーが費やした作業量および労力) をモニターするには、要求モニター・エレメントを使用します。

要求とは、データベース・リソースを消費する作業を実行するための、データベース・エージェントに対する命令です。要求のソースとしては、以下のようなものがあります。

- 外部アプリケーションにより直接出される命令。OPEN または EXECUTE 命令など。これらはアプリケーション要求と呼ばれます。
- コーディネーター・エージェントによって、同じまたは異なるデータベース・メンバーのサブエージェントに出される命令。
- 異なるデータベース・メンバーのエージェントにより出される命令。

要求モニター・エレメントは、さまざまなタイプの要求を処理するために、データベース・サーバーにより費やされた作業量または処理動作を測定します。これにはシステム全体の処理、特定のタイプの処理に関係した要求、および特定のデータ・サーバー環境に関係した要求が含まれます。

システム全体の処理情報を測定するためのいくつかの代表的なモニター・エレメントには、以下のものがあります。

- **rqsts_completed_total** モニター・エレメントは、システムにより完了された数を測定します。
- **total_rqst_time** モニター・エレメントは、待機時間と処理時間を含む、データ・サーバーの要求により費やされた時間を測定します。
- **total_wait_time** モニター・エレメントは、待機時間全体を測定します。
- **total_cpu_time** モニター・エレメントは、CPU 使用時間を測定します。

クライアント/サーバー処理情報を測定するためのいくつかの代表的なモニター・エレメントには、以下のものがあります。

- **client_idle_wait_time** モニター・エレメントは、オープン接続からの次の要求を待機するために費やされた時間を測定します。
- **tcpip_recv_volume** モニター・エレメントは、データ・サーバーがクライアントから TCP/IP 経由で受信したデータの量を測定します。

共通データ・サーバー処理操作を測定するためのいくつかの代表的なモニター・エレメントには、以下のものがあります。

- **pool_data_l_reads** は、バッファー・プール・リソース使用に関する情報を提供するモニター・エレメントの 1 つです。
- **pool_read_time** は、入出力処理に関する情報を提供するモニター・エレメントの 1 つです。
- **lock_wait_time** は、ロックおよび現在進行中のロックに関する情報を提供するモニター・エレメントの 1 つです。

- **total_section_sorts** は、ソートに関する情報を提供するモニター・エレメントの 1 つです。

選択したタイプのデータ・サーバー環境に関係したモニター処理のためのいくつかの代表的なモニター・エレメントには、以下のものがあります。

- **fcm_recv_wait_time** は、高速コミュニケーション・マネージャー (FCM) 処理を測定するモニター・エレメントの 1 つです。
- **wlm_queue_time_total** は、ワークロード管理制御アクションを測定するモニター・エレメントの 1 つです。

表関数を使用した要求メトリックへのアクセス

要求メトリックにアクセスするために、以下の表関数を使用することができます。

- MON_GET_SERVICE_SUBCLASS および MON_GET_SERVICE_SUBCLASS_DETAILS
- MON_GET_WORKLOAD および MON_GET_WORKLOAD_DETAILS
- MON_GET_CONNECTION および MON_GET_CONNECTION_DETAILS
- MON_GET_UNIT_OF_WORK および MON_GET_UNIT_OF_WORK_DETAILS

モニター表関数のこの一連の各表関数には 2 つの形式があり、その 1 つは名前の末尾に「DETAILS」が付きます。「DETAILS」が末尾に付かない関数は、最も一般的に必要とされるデータを返す SQL リレーショナル・インターフェースを提供します。それ以外の関数は、モニター・データへの XML ベースのアクセスを提供し、さらに総合的なデータ集合を返します。

この一連の表関数により、特定の集約レベルの要求メトリックに焦点を当てることができます。特定の状況で、関心対象のシステム・ワークロードのサブセット (または集約) に焦点を当てることのできる表関数を選択できます。これらのすべての表関数には、要求メトリック・モニター・エレメントの共通セットが組み込まれています。それぞれの表関数は、いくつかの追加の詳細を返す場合があり、それらはすべての表関数に共通というわけではありません。

ユーザー定義のワークロードまたはサービス・クラスがないデータベースでは、データベース・マネージャーによって実行されるすべてのユーザーの作業は、デフォルトのユーザー・ワークロードおよびユーザー・サービス・クラスで実行されます。各サービス・クラス (またはワークロード) にデータを返す表関数は、データベース全体のユーザー・ワークロードの処理を表す、単一のサービス・クラス (またはワークロード) のデータを返します。

ユーザー定義ワークロードおよびサービス・クラスがあるデータベースでは、各サービス・クラス (またはワークロード) のデータを返す表関数により、サービス・クラス (またはワークロード) 単位で処理を比較することができます。 SQL を使用すると、サービス・クラス (またはワークロード) 全体の値を合計して、データベース全体のユーザー・ワークロードの処理を表すモニター・エレメントの値を取得することができます。

イベント・モニターを使用した要求メトリックへのアクセス

要求メトリックは、以下のイベント・モニターにより報告されます。

- 統計イベント・モニター - 要求メトリックは、このイベント・モニターにより報告されるいくつかのタイプの情報の 1 つです。
- UoW イベント・モニター - このイベント・モニターは、MON_GET_UNIT_OF_WORK 表関数と類似のまたは同じフィールドを報告します。

第 9 章 アクティビティー・モニター・エレメント

アクティビティー・モニター・エレメントは、要求モニター・エレメントのサブセットです。アクティビティー・メトリックを使用して、アクティビティーの実行 (特に SQL ステートメント・セクションの実行のための処理) に関係した、データ・サーバー処理のサブセットをモニターします。

要求モニター・エレメントは、アプリケーション要求を処理するためにデータ・サーバーにより費やされた作業量および労力全体をモニターします。アクティビティー・モニター・エレメントは、ロック、ソート、および行処理を含む SQL ステートメント・セクションを実行するために行われる作業をモニターします。

アクティビティー・モニター・エレメントの現行値にアクセスするには、以下の表関数を使用します。

MON_GET_ACTIVITY_DETAILS
進行中の 1 つ以上のアクティビティーに関する詳細を返します。入力パラメーターに関心対象のアクティビティーを指定します。返されるデータには、アクティビティー・メトリック・モニター・エレメント、他の多くのモニター・エレメント、およびステートメント・テキストが含まれます。データは XML 形式で返されます。

MON_GET_PKG_CACHE_STMT
データベース・パッケージ・キャッシュ内の一部またはすべての SQL ステートメント・セクションの詳細を返します。これには静的および動的 SQL ステートメントの両方が含まれます。返されるデータには、パッケージ・キャッシュに追加されてからの、セクションのすべての実行を対象として集約されたアクティビティー・メトリック・モニター・エレメントが含まれます。データはリレーショナル形式で返されます。

アクティビティー・イベント・モニターを使用して、アクティビティーに関する履歴データにアクセスします。このモニターは、それぞれのアクティビティーの各実行データをキャプチャーします。アクティビティー・イベント・モニターは、MON_GET_ACTIVITY_DETAILS 表関数と同じアクティビティー・モニター・エレメントをキャプチャーします。これは、一部の追加情報もキャプチャーします。

第 10 章 データ・オブジェクトに関するモニター・エレメント

データ・オブジェクト・モニター・エレメントは、表、索引、バッファー・プール、表スペース、およびコンテナーを含む、特定のデータ・オブジェクトに対して実行された操作に関する情報を提供します。

すべてのデータ・オブジェクト・タイプには、モニター可能な一連のモニター・エレメントがあります。例えば、バッファー・プールには、バッファー・プール・ヒット率を計算するために使用できるエレメントがあります。

以下の表関数を使用して、データ・オブジェクト・モニター・エレメントの現行値にアクセスします。これらのモニター表関数は、リレーショナル形式でデータを返します。

- MON_GET_BUFFERPOOL
- MON_GET_TABLESPACE
- MON_GET_CONTAINER
- MON_GET_TABLE
- MON_GET_INDEX

第 11 章 作業単位イベント・モニターにより報告されるモニター・エレメント

以下のモニター・エレメントは、作業単位イベント・モニターにより報告されます。

- 364 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』
- 377 ページの『appl_id - アプリケーション ID :』
- 381 ページの『appl_name アプリケーション名 : モニター・エレメント』
- 395 ページの『auth_id 許可 ID』
- 409 ページの『client_acctng - クライアント・アカウンティング・ストリング : モニター・エレメント』
- 409 ページの『client_applname - クライアント・アプリケーション名 : モニター・エレメント』
- 412 ページの『client_pid クライアント・プロセス ID』
- 412 ページの『client_platform クライアント・オペレーティング・プラットフォーム』
- 413 ページの『client_prdid クライアント製品およびバージョン ID : モニター・エレメント』
- 413 ページの『client_protocol クライアント通信プロトコル』
- 414 ページの『client_userid - クライアントのユーザー ID : モニター・エレメント』
- 415 ページの『client_wrkstnname - クライアント・ワークステーション名 : モニター・エレメント』
- 418 ページの『completion_status 完了状況 : モニター・エレメント』
- 428 ページの『conn_time データベース接続時刻 : モニター・エレメント』
- 441 ページの『coord_partition_num コーディネーター・パーティション番号 : モニター・エレメント』
- 451 ページの『db_conn_time データベース活動化タイム・スタンプ : モニター・エレメント』
- 723 ページの『service_class_id サービス・クラス ID : モニター・エレメント』
- 724 ページの『service_subclass_name サービス・サブクラス名 : モニター・エレメント』
- 725 ページの『service_superclass_name サービス・スーパークラス名 : モニター・エレメント』
- 726 ページの『session_auth_id セッション許可 ID : モニター・エレメント』
- 828 ページの『uow_id 作業単位 ID : モニター・エレメント』
- 829 ページの『uow_start_time 作業単位開始タイム・スタンプ』
- 830 ページの『uow_stop_time 作業単位停止タイム・スタンプ : モニター・エレメント』

- 840 ページの『workload_id ワークロード ID : モニター・エレメント』
- 841 ページの『workload_name ワークロード名 : モニター・エレメント』
- 842 ページの『workload_occurrence_id ワークロード・オカレンス ID : モニター・エレメント』

第 12 章 ロック・イベント・モニターにより報告されるモニター・エレメント

これらのモニター・エレメントは、ロック・イベント・モニターにより報告されます。

- 553 ページの『lock_name ロック名 : モニター・エレメント』
- 555 ページの『lock_object_type - 待機中のロック対象タイプ : モニター・エレメント』
- 556 ページの『lock_release_flags ロック保留解除フラグ : モニター・エレメント』
- 557 ページの『lock_status - ロック状況 : モニター・エレメント』
- 557 ページの『lock_timeout_val ロック・タイムアウト値 : モニター・エレメント』
- 608 ページの『package_name - パッケージ名 : モニター・エレメント』
- 609 ページの『package_schema - パッケージ・スキーマ : モニター・エレメント』
- 609 ページの『package_version_id - パッケージ・バージョン : モニター・エレメント』
- 703 ページの『rolled_back_participant_no ロールバック参加アプリケーション : モニター・エレメント』
- 716 ページの『section_number - セクション番号 : モニター・エレメント』
- 723 ページの『service_class_id サービス・クラス ID : モニター・エレメント』
- 724 ページの『service_subclass_name サービス・サブクラス名 : モニター・エレメント』
- 749 ページの『stmt_invocation_id ステートメント呼び出し ID : モニター・エレメント』
- 750 ページの『stmt_lock_timeout ステートメント・ロック・タイムアウト : モニター・エレメント』
- 751 ページの『stmt_nest_level ステートメント・ネスト・レベル : モニター・エレメント』
- 751 ページの『stmt_operation/operation ステートメント操作 : モニター・エレメント』
- 753 ページの『stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID』
- 754 ページの『stmt_query_id ステートメント照会 ID : モニター・エレメント』
- 755 ページの『stmt_source_id ステートメント・ソース ID』
- 757 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』
- 758 ページの『stmt_type ステートメント・タイプ : モニター・エレメント』
- 759 ページの『stmt_value_data 値データ』
- 760 ページの『stmt_value_index 値索引』
- 760 ページの『stmt_value_isnull NULL 値の値 : モニター・エレメント』
- 761 ページの『stmt_value_isreopt ステートメント再最適化に使用される変数 : モニター・エレメント』
- 761 ページの『stmt_value_type 値タイプ : モニター・エレメント』
- 766 ページの『table_name - 表名 : モニター・エレメント』
- 768 ページの『table_schema - 表スキーマ名 : モニター・エレメント』
- 776 ページの『tablespace_name - 表スペース名 : モニター・エレメント』

- 828 ページの『uow_id 作業単位 ID : モニター・エレメント』
- 840 ページの『workload_id ワークロード ID : モニター・エレメント』
- 841 ページの『workload_name ワークロード名 : モニター・エレメント』

第 13 章 待機時間に関するモニター・エレメント

要求処理中に要求が待機のために時間を費やす条件および箇所を検出することは、パフォーマンス低下またはボトルネックの発生箇所を検出する効果的な方法です。待機時間モニター・エレメントは、特定のエンティティーまたはリソースの待機に費やされる時間を測定します。

例えば、パフォーマンス低下の一般的な説明として、「要求が特定のロックの待機に時間を費やしすぎている」と言われることがあります。遅延の性質を判別するために、**lock_wait_time** モニター・エレメントの値を考察できます。

それぞれの待機時間モニター・エレメントは、一連の要求モニター・エレメント、アクティビティー・モニター・エレメント、またはデータ・オブジェクト・モニター・エレメントにも属します。

表 70. 待機時間に関するモニター・エレメント

カテゴリー	サブカテゴリー	待機時間に関するモニター・エレメント
高位待機時間	DB2 処理中の待機時間。アクティビティーの実行を含む	818 ページの『total_wait_time - 合計待機時間 : モニター・エレメント』
	アクティビティー実行中のみの待機時間	801 ページの『total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント』
入出力	バッファー・プール入出力	649 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』 663 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』
	LOB の直接入出力	467 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』 471 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』
	トランザクション・ログ入出力	565 ページの『log_buffer_wait_time - ログ・バッファー待機時間 : モニター・エレメント』
	診断メッセージ・ログ	463 ページの『diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間 : モニター・エレメント』
ロック	-	560 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

表 70. 待機時間に関するモニター・エレメント (続き)

カテゴリー	サブカテゴリー	待機時間に関するモニター・エレメント
接続	エージェント待機時間	368 ページの『agent_wait_time - エージェント待機時間 : モニター・エレメント』
監査	-	390 ページの『audit_file_write_wait_time - 監査ファイル書き込み待機時間 : モニター・エレメント』 392 ページの『audit_subsystem_wait_time - 監査サブシステム待機時間 : モニター・エレメント』
FCM	FCM 全体 (要求メッセージおよび表キュー・データを含む)	494 ページの『fcm_send_wait_time - FCM 送信待機時間 : モニター・エレメント』 491 ページの『fcm_recv_wait_time - FCM 受信待機時間 : モニター・エレメント』
	要求メッセージに関連した FCM	488 ページの『fcm_message_send_wait_time - FCM メッセージの送信待機時間 : モニター・エレメント』 486 ページの『fcm_message_recv_wait_time - FCM メッセージの受信待機時間 : モニター・エレメント』
	表キューに関連した FCM	500 ページの『fcm_tq_send_wait_time - FCM 表キュー送信待機時間 : モニター・エレメント』 497 ページの『fcm_tq_recv_wait_time - FCM 表キュー受信待機時間 : モニター・エレメント』
ワークロード・マネージャー制御アクション	並行性しきい値の超過によるキューイング	837 ページの『wlm_queue_time_total - ワークロード・マネージャー合計キュー時間 : モニター・エレメント』

表 70. 待機時間に関するモニター・エレメント (続き)

カテゴリー	サブカテゴリー	待機時間に関するモニター・エレメント
クライアント/サーバー処理	アクティブ接続からの次の要求を待機した時間	411 ページの『client_idle_wait_time - クライアントのアイドル待機時間 : モニター・エレメント』
	リモート・クライアント・ネットワーク (TCPIP)	790 ページの『tcpip_recv_wait_time - TCP/IP 受信待機時間 : モニター・エレメント』 793 ページの『tcpip_send_wait_time - TCP/IP 送信待機時間 : モニター・エレメント』
	ローカル・クライアント通信 (IPC)	537 ページの『ipc_recv_wait_time - プロセス間通信受信待機時間 : モニター・エレメント』 539 ページの『ipc_send_wait_time - プロセス間通信送信待機時間 : モニター・エレメント』

待機時間エレメントは、以下のインターフェースから使用可能です。

- MON_GET_SERVICE_SUBCLASS 表関数
- MON_GET_SERVICE_SUBCLASS_DETAILS 表関数
- MON_GET_WORKLOAD 表関数
- MON_GET_WORKLOAD_DETAILS 表関数
- MON_GET_CONNECTION 表関数
- MON_GET_CONNECTION_DETAILS 表関数
- MON_GET_UNIT_OF_WORK 表関数
- MON_GET_UNIT_OF_WORK_DETAILS 表関数
- MON_GET_PKG_CACHE_STMT 表関数
- MON_GET_ACTIVITY_DETAILS 表関数
- 統計イベント・モニター (event_wlstats および event_scstats 論理グループ内の DETAILS_XML エレメント)
- アクティビティー・イベント・モニター (event_activity 論理グループ内の DETAILS_XML エレメント)
- 作業単位イベント・モニター

すべての待機時間エレメントが、すべてのインターフェースによって報告されるというわけではありません。例えば、**client_idle_wait_time** モニター・エレメントは、MON_GET_SERVICE_SUBCLASS 表関数などのシステム・レベルのインターフェースにのみ当てはまります。これは、データ・オブジェクト・モニター表関数では報告されません。

エレメントを報告するインターフェースのリストの、各モニター・エレメントの説明を参照してください。

第 14 章 論理データ・グループ

スナップショット・モニター・インターフェースの論理データ・グループへのマッピング

次の表では、スナップショット・モニター・データにアクセスするためのいくつかの方法をリストしています。すべてのスナップショット・モニター・データは、モニター・エレメント内に保管され、論理データ・グループによってカテゴリー化されます。それぞれの API 要求タイプ、CLP コマンド、および SQL 管理ビューは、すべての論理データ・グループのサブセットからのモニター・データのみをキャプチャーします。

この表にリストされている、それぞれの API 要求タイプ、CLP コマンド、および SQL 管理ビューは、右端の列にリストされている論理データ・グループからのモニター・エレメントを戻します。

注:

1. 対応する SQL 管理ビューがない、いくつかの API 要求タイプおよび CLP コマンドがあります。その他の API 要求タイプおよび CLP コマンドの場合、それぞれの SQL 管理ビューは、関連した論理データ・グループのサブセットをキャプチャーします。
2. 一部のモニター・エレメントは、関連したモニター・スイッチが ON に設定されている場合にだけ戻されます。スイッチで必須エレメントを制御できるかどうか判別するには、個々のモニター・エレメントを参照してください。

表 71. スナップショット・モニター・インターフェースの論理データ・グループへのマッピング

db2GetSnapshot API 要求タイプ	CLP コマンド	SQL 管理ビュー	論理データ・グループ
SQLMA_APPLINFO_ALL	`list applications [show detail]`	APPLICATIONS	appl_info
SQLMA_DBASE_APPLINFO	`list applications for database` *dbname* `[show detail]`	APPLICATIONS	appl_info
SQLMA_DCS_APPLINFO_ALL	`list dcs applications [show detail]`		dcs_appl_info

表 71. スナップショット・モニター・インターフェースの論理データ・グループへのマッピング (続き)

db2GetSnapshot API 要求タイプ	CLP コマンド	SQL 管理ビュー	論理データ・グループ
SQLMA_DB2	get snapshot for dbm	SNAPDBM	db2
		SNAPFCM	fcm
		SNAPFCMPART	fcm_node
		SNAPUTIL	utility_info
		SNAPUTIL_PROGRESS	progress、progress_info
		SNAPDBM_MEMORY_POOL	memory_pool
	get dbm monitor switches	SNAPSWITCHES	switch_list
SQLMA_DBASE	get snapshot for database on *dbname*	SNAPDB	dbase
		SNAPDETAILLOG	detail_log
		SNAPSTORAGE_PATHS	db_storage_group
			rollforward
			db_sto_path_info
		SNAPTBSP	tablespace
		SNAPDB_MEMORY_POOL	memory_pool
SQLMA_DBASE_ALL	get snapshot for all databases	SNAPDB	dbase
		SNAPSTORAGE_PATHS	db_storage_group
			rollforward
			db_sto_path_info
		SNAPTBSP	tablespace
		SNAPDB_MEMORY_POOL	memory_pool
	list active databases		dbase
SQLMA_DCS_DBASE	get snapshot for dcs database on *dbname*		dcs_dbase、stmt_transmissions
SQLMA_DCS_DBASE_ALL	get snapshot for all dcs databases		dcs_dbase、stmt_transmissions
SQLMA_DBASE_REMOTE	get snapshot for remote database on *dbname*		dbase_remote
SQLMA_DBASE_REMOTE_ALL	get snapshot for all remote databases		dbase_remote
SQLMA_APPL	get snapshot for application applid *appl-id*	SNAPAPPL	appl
		SNAPAGENT	agent
		SNAPAPPL_INFO	appl_info
		SNAPLOCKWAIT	lock_wait
		SNAPSTMT	stmt
		SNAPSUBSECTION	サブセクション
		SNAPAGENT_MEMORY_POOL	memory_pool

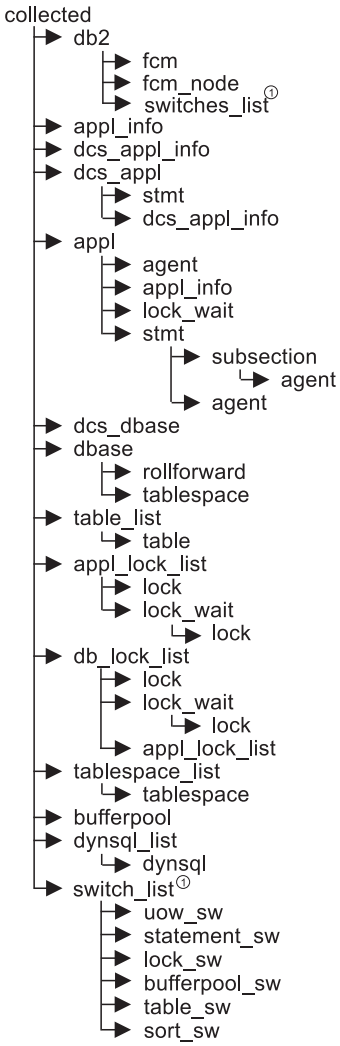
表 71. スナップショット・モニター・インターフェースの論理データ・グループへのマッピング (続き)

db2GetSnapshot API 要求タイプ	CLP コマンド	SQL 管理ビュー	論理データ・グループ
SQLMA_AGENT_ID	get snapshot for application agentid *appl-handle*	SNAPAGENT	appl
		SNAPAGENT	agent
		SNAPAPPL_INFO	appl_info
		SNAPLOCKWAIT	lock_wait
		SNAPSTMT	stmt
		SNAPSUBSECTION	サブセクション
		SNAPAGENT_MEMORY_POOL	memory pool
SQLMA_DBASE_APPLS	get snapshot for applications on *dbname*	SNAPAPPL	appl
		SNAPAGENT	agent
		SNAPAPPL_INFO	appl_info
		SNAPLOCKWAIT	lock_wait
		SNAPSTMT	stmt
		SNAPSUBSECTION	サブセクション
		SNAPAGENT_MEMORY_POOL	memory_pool
SQLMA_APPL_ALL	get snapshot for all applications	SNAPAPPL	appl
		SNAPAPPL_INFO	appl_info
		SNAPLOCKWAIT	lock_wait
		SNAPSTATEMENT	stmt
		SNAPAGENT	agent
		SNAPSUBSECTION	サブセクション
		SNAPAGENT_MEMORY_POOL	memory_pool
SQLMA_DCS_APPL	get snapshot for dcs application applid *appl-id*		dcs_appl、dcs_stmt、dcs_appl_info、stmt_transmissions
SQLMA_DCS_APPL_ALL	get snapshot for all dcs applications		dcs_appl、dcs_stmt、dcs_appl_info、stmt_transmissions
SQLMA_DCS_APPL_HANDLE	get snapshot for dcs application agentid *appl-handle*		dcs_appl、dcs_stmt、dcs_appl_info、stmt_transmissions
SQLMA_DCS_DBASE_APPLS	get snapshot for dcs applications on *dbname*		dcs_appl、dcs_stmt、dcs_appl_info、stmt_transmissions
SQLMA_DBASE_APPLS_REMOTE	get snapshot for remote applications on *dbname*		dbase_appl
SQLMA_APPL_REMOTE_ALL	get snapshot for all remote applications		dbase_appl

表 71. スナップショット・モニター・インターフェースの論理データ・グループへのマッピング (続き)

db2GetSnapshot API 要求タイプ	CLP コマンド	SQL 管理ビュー	論理データ・グループ
SQLMA_DBASE_TABLES	get snapshot for tables on *dbname*	SNAPTAB	table
		SNAPTAB_REORG	table_reorg
			table_list
SQLMA_APPL_LOCKS	get snapshot for locks for application applid *appl-id*	SNAPLOCK、SNAPAPPL、SNAPLOCKWAIT	appl_lock_list、lock_wait, lock
SQLMA_APPL_LOCKS_AGENT_ID	get snapshot for locks for application agentid *appl-handle*	SNAPLOCK、SNAPAPPL、SNAPLOCKWAIT	appl_lock_list、lock_wait, lock
SQLMA_DBASE_LOCKS	get snapshot for locks on *dbname*	SNAPLOCK	appl_lock_list、lock
		SNAPLOCK、SNAPLOCKWAIT	db_lock_list、lock_wait
SQLMA_DBASE_TABLESPACES	get snapshot for tablespaces on *dbname*	SNAPTBSP	tablespace
		SNAPTBSPPART	tablespace、tablespace_nodeinfo
		SNAPTBSP_QUIESCER	tablespace_quiescer、tablespace_nodeinfo
		SNAPCONTAINER	tablespace_container、tablespace_nodeinfo
		SNAPTBSP_RANGE	tablespace_ranges、tablespace_nodeinfo
			tablespace_list、tablespace_nodeinfo
SQLMA_BUFFERPOOLS_ALL	get snapshot for all bufferpools	SNAPBP	バッファー・プール
SQLMA_DBASE_BUFFERPOOLS	get snapshot for bufferpools on *dbname*	SNAPBP	バッファー・プール
SQLMA_DYNAMIC_SQL	get snapshot for dynamic sql on *dbname*	SNAPDYN_SQL	dynsql
			dynsql_list

以下の図は、論理データ・グループがスナップショット・データ・ストリーム内に現れる順番を示しています。



① 類似した構造 (下位の level_sw 項目が db2 により戻されるが、図には示されていない)

図 8. データ・ストリーム階層

注: 論理データ・グループの一部として、時間が戻されることがあります。

スナップショット・モニターの論理データ・グループおよびモニター・エレメント

次のセクションでは、論理データ・グループと、スナップショット・モニターによって戻されるモニター・エレメントをリストしています。

- 282 ページの『agent 論理データ・グループ』
- 283 ページの『appl 論理データ・グループ』
- 286 ページの『appl_id_info 論理データ・グループ』
- 287 ページの『appl_info 論理データ・グループ』
- 288 ページの『appl_lock_list 論理データ・グループ』
- 288 ページの『appl_remote 論理データ・グループ』
- 289 ページの『bufferpool 論理データ・グループ』

- 291 ページの『bufferpool_nodeinfo 論理データ・グループ』
- 291 ページの『collected 論理データ・グループ』
- 291 ページの『db2 論理データ・グループ』
- 292 ページの『db_lock_list 論理データ・グループ』
- 292 ページの『dbase 論理データ・グループ』
- 298 ページの『dbase_remote 論理データ・グループ』
- 298 ページの『db_storage_group 論理データ・グループ』
- 299 ページの『dcs_appl 論理データ・グループ』
- 301 ページの『dcs_appl_info 論理データ・グループ』
- 301 ページの『dcs_dbase 論理データ・グループ』
- 303 ページの『dcs_stmt 論理データ・グループ』
- 304 ページの『detail_log 論理データ・グループ』
- 304 ページの『dynsql 論理データ・グループ』
- 306 ページの『dynsql_list 論理データ・グループ』
- 306 ページの『fcm 論理データ・グループ』
- 306 ページの『fcm_node 論理データ・グループ』
- 306 ページの『hadr 論理データ・グループ』
- 307 ページの『lock 論理データ・グループ』
- 307 ページの『lock_wait 論理データ・グループ』
- 308 ページの『memory_pool 論理データ・グループ』
- 308 ページの『progress 論理データ・グループ』
- 308 ページの『progress_list 論理データ・グループ』
- 308 ページの『rollforward 論理データ・グループ』
- 309 ページの『stmt 論理データ・グループ』
- 310 ページの『stmt_transmissions 論理データ・グループ』
- 312 ページの『subsection 論理データ・グループ』
- 312 ページの『table 論理データ・グループ』
- 313 ページの『table_list 論理データ・グループ』
- 313 ページの『table_reorg 論理データ・グループ』
- 314 ページの『tablespace 論理データ・グループ』
- 316 ページの『tablespace_container 論理データ・グループ』
- 316 ページの『tablespace_list 論理データ・グループ』
- 316 ページの『tablespace_nodeinfo 論理データ・グループ』
- 317 ページの『tablespace_quiescer 論理データ・グループ』
- 317 ページの『tablespace_range 論理データ・グループ』
- 318 ページの『utility_info 論理データ・グループ』

agent 論理データ・グループ

366 ページの『agent_pid エンジン・ディスパッチ可能単位 (EDU) ID : モニター・エレメント』

appl 論理データ・グループ

728 ページの『shr_workspace_section_lookups 共有ワークスペース・セクション検索』

728 ページの『shr_workspace_size_top 最大共有ワークスペース・サイズ』

731 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

735 ページの『sql_reqs_since_commit 最終コミット後の SQL 要求』

744 ページの『static_sql_stmts 試行された静的 SQL ステートメント』

805 ページの『total_hash_joins ハッシュ結合の合計』

806 ページの『total_hash_loops ハッシュ・ループの合計』

807 ページの『total_olap_funcs OLAP 関数の合計数 : モニター・エレメント』

813 ページの『total_sort_time - ソート時間合計 : モニター・エレメント』

814 ページの『total_sorts - ソート合計 : モニター・エレメント』

825 ページの『uid_sql_stmts 実行された更新/挿入/削除 SQL ステートメント』

826 ページの『unread_prefetch_pages - 読み取り不能プリフェッチ・ページ : モニター・エレメント』

827 ページの『uow_comp_status 作業単位完了状況』

827 ページの『uow_elapsed_time 最新の作業単位の経過時間』

828 ページの『uow_lock_wait_time - ロック待機中の作業単位の合計時間 : モニター・エレメント』

829 ページの『uow_log_space_used 使用されている作業単位ログ・スペース』

829 ページの『uow_start_time 作業単位開始タイム・スタンプ』

830 ページの『uow_stop_time 作業単位停止タイム・スタンプ : モニター・エレメント』

843 ページの『x_lock_escals 排他ロック・エスカレーション数』

845 ページの『xquery_stmts - 試行された XQuery ステートメント』

appl_id_info 論理データ・グループ

364 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

377 ページの『appl_id - アプリケーション ID :』

381 ページの『appl_name アプリケーション名 : モニター・エレメント』

384 ページの『appl_status アプリケーション状況』

395 ページの『auth_id 許可 ID』

410 ページの『client_db_alias アプリケーションで使用するデータベース別名』

413 ページの『client_prdid クライアント製品およびバージョン ID : モニター・エレメント』

416 ページの『codepage_id アプリケーションで使用するコード・ページ ID』

452 ページの『db_name データベース名』

453 ページの『db_path データベース・パス』

528 ページの『input_db_alias 入力データベース別名』

719 ページの『sequence_no シーケンス番号 : モニター・エレメント』

747 ページの『status_change_time アプリケーション状況変更時刻』

appl_info 論理データ・グループ

364 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

377 ページの『appl_id - アプリケーション ID :』

381 ページの『appl_name アプリケーション名 : モニター・エレメント』

383 ページの『appl_section_inserts セクション挿入数 : モニター・エレメント』

383 ページの『appl_section_lookups - セクション検索』

384 ページの『appl_status アプリケーション状況』

395 ページの『auth_id 許可 ID』

395 ページの『authority_bitmap ユーザー許可レベル : モニター・エレメント』

396 ページの『authority_lvl ユーザー許可レベル : モニター・エレメント』

410 ページの『client_db_alias アプリケーションで使用するデータベース別名』

412 ページの『client_pid クライアント・プロセス ID』

412 ページの『client_platform クライアント・オペレーティング・プラットフォーム』

413 ページの『client_prdid クライアント製品およびバージョン ID : モニター・エレメント』

413 ページの『client_protocol クライアント通信プロトコル』

416 ページの『codepage_id アプリケーションで使用するコード・ページ ID』

440 ページの『coord_agent_pid コーディネーター・エージェント ID : モニター・エレメント』

441 ページの『coord_node コーディネーター・ノード』

442 ページの『corr_token DRDA 相関トークン』

452 ページの『db_name データベース名』

453 ページの『db_path データベース・パス』

484 ページの『execution_id ユーザー・ログイン ID』

528 ページの『input_db_alias 入力データベース別名』

541 ページの『is_system_appl システム・アプリケーション : モニター・エレメント』

591 ページの『num_assoc_agents 関連したエージェント数』

719 ページの『sequence_no シーケンス番号 : モニター・エレメント』

726 ページの『session_auth_id セッション許可 ID : モニター・エレメント』

747 ページの『status_change_time アプリケーション状況変更時刻』

795 ページの『territory_code データベース・テリトリー・コード』

818 ページの『tpmon_acc_str TP モニター・クライアント・アカウンティング・ストリング : モニター・エレメント』

819 ページの『tpmon_client_app TP モニター・クライアント・アプリケーション名 : モニター・エレメント』

820 ページの『tpmon_client_userid TP モニター・クライアント・ユーザー ID : モニター・エレメント』

820 ページの『tpmon_client_wkstn TP モニター・クライアント・ワークステーション名 : モニター・エレメント』

840 ページの『workload_id ワークロード ID : モニター・エレメント』

appl_lock_list 論理データ・グループ

364 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

377 ページの『appl_id - アプリケーション ID :』

381 ページの『appl_name アプリケーション名 : モニター・エレメント』

384 ページの『appl_status アプリケーション状況』

395 ページの『auth_id 許可 ID』

410 ページの『client_db_alias アプリケーションで使用するデータベース別名』

416 ページの『codepage_id アプリケーションで使用するコード・ページ ID』

560 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

564 ページの『locks_held ロック保持数』

565 ページの『locks_waiting ロックで待機中の現行エージェント』

719 ページの『sequence_no シーケンス番号 : モニター・エレメント』

726 ページの『session_auth_id セッション許可 ID : モニター・エレメント』

747 ページの『status_change_time アプリケーション状況変更時刻』

appl_remote 論理データ・グループ

417 ページの『commit_sql_stmts 試行されたコミット・ステートメント』

445 ページの『create_nickname ニックネーム作成回数』

446 ページの『create_nickname_time ニックネーム作成応答時間』

450 ページの『datasource_name データ・ソース名』

452 ページの『db_name データベース名』

462 ページの『delete_sql_stmts 削除回数』

462 ページの『delete_time 削除応答時間』

484 ページの『failed_sql_stmts 失敗したステートメント操作』

528 ページの『insert_sql_stmts 挿入回数』

529 ページの『insert_time 挿入応答時間』

615 ページの『passthru_time パススルー時間』

616 ページの『passthrus パススルー数』

692 ページの『remote_lock_time リモート・ロック時間』

693 ページの『remote_locks リモート・ロック数』

701 ページの『rollback_sql_stmts 試行されたロールバック・ステートメント』

704 ページの『rows_deleted - 削除行数 : モニター・エレメント』

705 ページの『rows_inserted - 挿入行数 : モニター・エレメント』

711 ページの『rows_selected 選択行数』

712 ページの『rows_updated - 更新行数 : モニター・エレメント』

718 ページの『select_sql_stmts 実行された選択 SQL ステートメント』

719 ページの『select_time 照会応答時間』

734 ページの『sp_rows_selected ストアード・プロシージャーによって戻された行数』

763 ページの『stored_proc_time ストアード・プロシージャー時間』

763 ページの『stored_procs ストアード・プロシージャー数』

831 ページの『update_sql_stmts 更新回数』

832 ページの『update_time 更新応答時間』

bufferpool 論理データ・グループ

399 ページの『block_ios - ブロック入出力要求数 : モニター・エレメント』

401 ページの『bp_id バッファー・プール ID : モニター・エレメント』

402 ページの『bp_name - バッファー・プール名 : モニター・エレメント』

452 ページの『db_name データベース名』

453 ページの『db_path データベース・パス』

465 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

467 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

468 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

470 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

471 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

473 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

502 ページの『files_closed - クローズしたデータベース・ファイル : モニター・エレメント』

528 ページの『input_db_alias 入力データベース別名』

611 ページの『pages_from_block_ios - ブロック入出力によって読み取られたページ数の合計 : モニター・エレメント』

611 ページの『pages_from_vectored_ios - ベクトル化入出力によって読み取られたページ数の合計 : モニター・エレメント』

621 ページの『pool_async_data_read_reqs - バッファー・プール非同期読み取り要求 : モニター・エレメント』

622 ページの『pool_async_data_reads バッファー・プール非同期データ読み取り : モニター・エレメント』

623 ページの『pool_async_data_writes - バッファー・プール非同期データ書き込み : モニター・エレメント』

624 ページの『pool_async_index_read_reqs - バッファー・プール非同期索引読み取り要求 : モニター・エレメント』

624 ページの『pool_async_index_reads - バッファー・プール非同期索引読み取り : モニター・エレメント』

625 ページの『pool_async_index_writes - バッファー・プール非同期索引書き込み : モニター・エレメント』

626 ページの『pool_async_read_time バッファー・プール非同期読み取り時間』

627 ページの『pool_async_write_time バッファー・プール非同期書き込み時間』

628 ページの『pool_async_xda_read_reqs - バッファー・プール非同期 XDA 読み取り要求 : モニター・エレメント』

629 ページの『pool_async_xda_reads - バッファー・プール非同期 XDA データ読み取り : モニター・エレメント』

630 ページの『pool_async_xda_writes - バッファー・プール非同期 XDA データ書き込み : モニター・エレメント』

632 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

634 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

635 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

641 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

643 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

645 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

648 ページの『pool_no_victim_buffer - バッファー・プールの非ビクティム・バッファー数 : モニター・エレメント』

649 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

652 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

654 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

655 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

657 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

659 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

661 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

663 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

665 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

667 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

669 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』

836 ページの『vectored_ios - ベクトル化入出力要求数 : モニター・エレメント』

bufferpool_nodeinfo 論理データ・グループ

401 ページの『bp_cur_buffsz バッファー・プールの現行サイズ』

402 ページの『bp_new_buffsz 新規バッファー・プール・サイズ』

402 ページの『bp_pages_left_to_remove 除去残ページ数』

403 ページの『bp_tbsp_use_count バッファー・プールにマップされている表スペースの数』

590 ページの『node_number ノード番号』

collected 論理データ・グループ

590 ページの『node_number ノード番号』

720 ページの『server_db2_type モニター対象 (サーバー) ノードのデータベース・マネージャーのタイプ』

721 ページの『server_instance_name サーバー・インスタンス名』

722 ページの『server_prdid - サーバー製品/バージョン ID』

722 ページの『server_version サーバー・バージョン』

799 ページの『time_stamp スナップショット時刻』

800 ページの『time_zone_disp 時間帯変位』

db2 論理データ・グループ

371 ページの『agents_created_empty_pool エージェント・プールが空のために作成されたエージェント』

371 ページの『agents_from_pool - プールから割り当てられたエージェント :』

372 ページの『agents_registered 登録済みエージェント』

372 ページの『agents_registered_top - エージェント最大登録数 :』

372 ページの『agents_stolen スチールされたエージェント』

373 ページの『agents_waiting_on_token - トークン待ちエージェント :』

374 ページの『agents_waiting_top - エージェント最大待機数 : モニター・エレメント』

417 ページの『comm_private_mem - コミット済み専用メモリー :』

419 ページの『con_local_dbases 現行接続を持つローカル・データベース』

441 ページの『coord_agents_top - コーディネーター・エージェント最大数 :』

451 ページの『db2start_time データベース・マネージャー開始タイム・スタンプ』

454 ページの『db_status データベース状況』

508 ページの『gw_cons_wait_client クライアントの要求送信を待機している接続の数』

509 ページの『gw_cons_wait_host ホストの応答を待機している接続の数』

509 ページの『gw_cur_cons DB2 Connect の現在の接続数』

510 ページの『gw_total_cons DB2 Connect の接続試行合計回数』

525 ページの『idle_agents - アイドル・エージェント数 :』

543 ページの『last_reset 最後のリセット・タイム・スタンプ』

545 ページの『local_cons - ローカル接続 :』

545 ページの『local_cons_in_exec - データベース・マネージャーで実行中のローカル接続 :』

572 ページの『max_agent_overflows 最大エージェント・オーバーフロー回数 : モニター・エレメント』

594 ページの『num_gw_conn_switches - 接続切り替え回数』

597 ページの『num_nodes_in_db2_instance パーティション内のノード数』

616 ページの『piped_sorts_accepted 受け入れられたパイプ・ソート』

617 ページの『piped_sorts_requested 要求されたパイプ・ソート数』

673 ページの『post_threshold_hash_joins ハッシュ結合のしきい値』

673 ページの『post_threshold_olap_funcs OLAP 関数のしきい値 : モニター・エレメント』

674 ページの『post_threshold_sorts - ポストしきい値ソート : モニター・エレメント』

680 ページの『product_name 製品名』

691 ページの『rem_cons_in - データベース・マネージャーへのリモート接続 :』

692 ページの『rem_cons_in_exec - データベース・マネージャーで実行中のリモート接続 :』

724 ページの『service_level サービス・レベル』

729 ページの『smallest_log_avail_node 使用可能なログ・スペースが最小のノード』

729 ページの『sort_heap_allocated 割り振られたソート・ヒープの合計』

730 ページの『sort_heap_top ソート専用ヒープの最高水準点』

db_lock_list 論理データ・グループ

387 ページの『appls_cur_cons - 現在接続されているアプリケーション :』

452 ページの『db_name データベース名』

453 ページの『db_path データベース・パス』

528 ページの『input_db_alias 入力データベース別名』

564 ページの『locks_held ロック保持数』

565 ページの『locks_waiting ロックで待機中の現行エージェント』

dbase 論理データ・グループ

359 ページの『active_hash_joins - アクティブ・ハッシュ結合』

359 ページの『active_olap_funcs アクティブ OLAP 関数 : モニター・エレメント』

359 ページの『active_sorts アクティブ・ソート』

373 ページの『agents_top 作成されたエージェントの数』

380 ページの『appl_id_oldest_xact 最も古いトランザクションを持つアプリケーション』

383 ページの『appl_section_inserts セクション挿入数 : モニター・エレメント』

383 ページの『appl_section_lookups - セクション検索』

387 ページの『appls_cur_cons - 現在接続されているアプリケーション :』

388 ページの『appls_in_db2 - データベースで現在実行中のアプリケーション :』

389 ページの『async_runstats - 非同期 RUNSTATS 要求の合計数: モニター・エレメント』

398 ページの『binds_precompiles 試行されたバインド/プリコンパイル』

400 ページの『blocks_pending_cleanup クリーンアップ保留中のロールアウト済みブロック : モニター・エレメント』

404 ページの『cat_cache_inserts - カタログ・キャッシュ挿入数』

405 ページの『cat_cache_lookups カタログ・キャッシュ検索』

406 ページの『cat_cache_overflows カタログ・キャッシュ・オーバーフロー数』

407 ページの『cat_cache_size_top - カタログ・キャッシュの最高水準点 : モニター・エレメント』

407 ページの『catalog_node カタログ・ノード番号』

408 ページの『catalog_node_name カタログ・ノード・ネットワーク名』

417 ページの『commit_sql_stmts 試行されたコミット・ステートメント』

429 ページの『connections_top 同時接続の最大数』

441 ページの『coord_agents_top - コーディネーター・エージェント最大数 :』

451 ページの『db_conn_time データベース活動化タイム・スタンプ : モニター・エレメント』

452 ページの『db_heap_top 割り振られた最大データベース・ヒープ』

452 ページの『db_location データベース・ロケーション』

452 ページの『db_name データベース名』

453 ページの『db_path データベース・パス』

454 ページの『db_status データベース状況』

457 ページの『ddl_sql_stmts データ定義言語 (DDL) SQL ステートメント』

459 ページの『deadlocks - デッドロック検出数 : モニター・エレメント』

465 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

467 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

468 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

470 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

471 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

473 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

476 ページの『dynamic_sql_stmts 試行された動的 SQL ステートメント』

484 ページの『failed_sql_stmts 失敗したステートメント操作』

502 ページの『files_closed - クローズしたデータベース・ファイル : モニター・エレメント』

521 ページの『hash_join_overflows ハッシュ結合のオーバーフロー』

618 ページの『pkg_cache_inserts パッケージ・キャッシュ挿入』

618 ページの『pkg_cache_lookups パッケージ・キャッシュ検索』

620 ページの『pkg_cache_num_overflows パッケージ・キャッシュ・オーバーフロー』

620 ページの『pkg_cache_size_top パッケージ・キャッシュの最高水準点』

621 ページの『pool_async_data_read_reqs - バッファー・プール非同期読み取り要求 : モニター・エレメント』

622 ページの『pool_async_data_reads バッファー・プール非同期データ読み取り : モニター・エレメント』

623 ページの『pool_async_data_writes - バッファー・プール非同期データ書き込み : モニター・エレメント』

624 ページの『pool_async_index_read_reqs - バッファー・プール非同期索引読み取り要求 : モニター・エレメント』

624 ページの『pool_async_index_reads - バッファー・プール非同期索引読み取り : モニター・エレメント』

625 ページの『pool_async_index_writes - バッファー・プール非同期索引書き込み : モニター・エレメント』

626 ページの『pool_async_read_time バッファー・プール非同期読み取り時間』

627 ページの『pool_async_write_time バッファー・プール非同期書き込み時間』

628 ページの『pool_async_xda_read_reqs - バッファー・プール非同期 XDA 読み取り要求 : モニター・エレメント』

629 ページの『pool_async_xda_reads - バッファー・プール非同期 XDA データ読み取り : モニター・エレメント』

630 ページの『pool_async_xda_writes - バッファー・プール非同期 XDA データ書き込み : モニター・エレメント』

632 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

634 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

635 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

638 ページの『pool_drty_pg_steal_clns - 起動されたバッファー・プール・ビクティム・ページ・クリーナー : モニター・エレメント』

639 ページの『pool_drty_pg_thrsh_clns - 起動されたバッファー・プールしきい値クリーナー : モニター・エレメント』

641 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

643 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

645 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

647 ページの『pool_lsn_gap_clns - 起動されたバッファー・プール・ログ・スペース・クリーナー : モニター・エレメント』

718 ページの『select_sql_stmts 実行された選択 SQL ステートメント』

721 ページの『server_platform サーバーのオペレーティング・システム』

726 ページの『shr_workspace_num_overflows 共有ワークスペースのオーバーフロー回数』

727 ページの『shr_workspace_section_inserts 共有ワークスペース・セクション挿入数』

728 ページの『shr_workspace_section_lookups 共有ワークスペース・セクション検索』

728 ページの『shr_workspace_size_top 最大共有ワークスペース・サイズ』

729 ページの『sort_heap_allocated 割り振られたソート・ヒープの合計』

731 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

732 ページの『sort_shrheap_allocated 現在割り振られているソート共有ヒープ』

733 ページの『sort_shrheap_top ソート共有ヒープの最高水準点』

744 ページの『static_sql_stmts 試行された静的 SQL ステートメント』

745 ページの『stats_cache_size - 統計キャッシュのサイズ: モニター・エレメント』

746 ページの『stats_fabricate_time - 統計作成アクティビティーに費やされた合計時間: モニター・エレメント』

746 ページの『stats_fabrications - 統計作成の合計数: モニター・エレメント』

763 ページの『sync_runstats - 同期 RUNSTATS アクティビティーの合計数: モニター・エレメント』

764 ページの『sync_runstats_time - 同期 RUNSTATS アクティビティーに費やされた合計時間: モニター・エレメント』

800 ページの『tot_log_used_top 使用された最大合計ログ・スペース』

803 ページの『total_cons データベース活動化以降の接続』

805 ページの『total_hash_joins ハッシュ結合の合計』

806 ページの『total_hash_loops ハッシュ・ループの合計』

806 ページの『total_log_available 使用可能なログの合計』

807 ページの『total_log_used 使用されているログ・スペースの合計』

807 ページの『total_olap_funcs OLAP 関数の合計数 : モニター・エレメント』

809 ページの『total_sec_cons 2 次接続』

813 ページの『total_sort_time - ソート時間合計 : モニター・エレメント』

814 ページの『total_sorts - ソート合計 : モニター・エレメント』

825 ページの『uid_sql_stmts 実行された更新/挿入/削除 SQL ステートメント』

826 ページの『unread_prefetch_pages - 読み取り不能プリフェッチ・ページ : モニター・エレメント』

843 ページの『x_lock_escals 排他ロック・エスカレーション数』

845 ページの『xquery_stmts - 試行された XQuery ステートメント』

dbase_remote 論理データ・グループ

db_storage_group 論理データ・グループ

590 ページの『node_number ノード番号』

762 ページの『sto_path_free_sz - 自動ストレージ・パスのフリー・スペース』

dcs_appl 論理データ・グループ

381 ページの『appl_idle_time アプリケーション・アイドル時間』

417 ページの『commit_sql_stmts 試行されたコミット・ステートメント』

478 ページの『elapsed_exec_time ステートメント実行経過時間』

484 ページの『failed_sql_stmts 失敗したステートメント操作』

508 ページの『gw_con_time DB2 Connect ゲートウェイの最初の接続開始』

510 ページの『gw_exec_time DB2 Connect ゲートウェイ処理の経過時間』

524 ページの『host_response_time ホスト応答時間』

525 ページの『inbound_bytes_received 受信されたインバウンド・バイト数』

526 ページの『inbound_bytes_sent 送信されたインバウンド・バイト数』

543 ページの『last_reset 最後のリセット・タイム・スタンプ』

573 ページの『max_data_received_1024 受信アウトバウンド・バイト数が 513 から 1024 バイトのステートメント数』

573 ページの『max_data_received_128 受信アウトバウンド・バイト数が 1 から 128 バイトのステートメント数』

574 ページの『max_data_received_16384 受信アウトバウンド・バイト数が 8193 から 16384 バイトのステートメント数』

574 ページの『max_data_received_2048 受信アウトバウンド・バイト数が 1025 から 2048 バイトのステートメント数』

575 ページの『max_data_received_256 受信アウトバウンド・バイト数が 129 から 256 バイトのステートメント数』

575 ページの『max_data_received_31999 受信アウトバウンド・バイト数が 16385 から 31999 バイトのステートメント数 : モニター・エレメント』

576 ページの『max_data_received_4096 受信アウトバウンド・バイト数が 2049 から 4096 バイトのステートメント数』

576 ページの『max_data_received_512 受信アウトバウンド・バイト数が 257 から 512 バイトのステートメント数』

577 ページの『max_data_received_64000 受信アウトバウンド・バイト数が 32000 から 64000 バイトのステートメント数 : モニター・エレメント』

577 ページの『max_data_received_8192 受信アウトバウンド・バイト数が 4097 から 8192 バイトのステートメント数』

577 ページの『max_data_received_gt64000 受信アウトバウンド・バイト数が 64000 バイトを超えるステートメント数』

578 ページの『max_data_sent_1024 送信アウトバウンド・バイト数が 513 から 1024 バイトのステートメント数』

578 ページの『max_data_sent_128 送信アウトバウンド・バイト数が 1 から 128 バイトのステートメント数』

579 ページの『max_data_sent_16384 送信アウトバウンド・バイト数が 8193 から 16384 バイトのステートメント数』

579 ページの『max_data_sent_2048 送信アウトバウンド・バイト数が 1025 から 2048 バイトのステートメント数』

580 ページの『max_data_sent_256 送信アウトバウンド・バイト数が 129 から 256 バイトのステートメント数』

580 ページの『max_data_sent_31999 送信アウトバウンド・バイト数が 16385 から 31999 バイトのステートメント数』

581 ページの『max_data_sent_4096 送信アウトバウンド・バイト数が 2049 から 4096 バイトのステートメント数』

581 ページの『max_data_sent_512 送信アウトバウンド・バイト数が 257 から 512 バイトのステートメント数』

582 ページの『max_data_sent_64000 送信アウトバウンド・バイト数が 32000 から 64000 バイトのステートメント数』

582 ページの『max_data_sent_8192 送信アウトバウンド・バイト数が 4097 から 8192 バイトのステートメント数』

583 ページの『max_data_sent_gt64000 送信アウトバウンド・バイト数が 64000 バイトを超えるステートメント数』

583 ページの『max_network_time_100_ms ネットワーク時間が 16 から 100 ミリ秒のステートメント数』

584 ページの『max_network_time_16_ms ネットワーク時間が 4 から 16 ミリ秒のステートメント数』

584 ページの『max_network_time_1_ms ネットワーク時間が 1 ミリ秒以下のステートメント数』

585 ページの『max_network_time_4_ms ネットワーク時間が 1 から 4 ミリ秒のステートメント数』

585 ページの『max_network_time_500_ms ネットワーク時間が 100 から 500 ミリ秒のステートメント数』

586 ページの『max_network_time_gt500_ms ネットワーク時間が 500 ミリ秒を超えるステートメント数』

588 ページの『network_time_bottom ステートメントの最小ネットワーク時間』

589 ページの『network_time_top ステートメントの最大ネットワーク時間』

600 ページの『open_cursors オープン・カーソル数』

604 ページの『outbound_bytes_received 受信されたアウトバウンド・バイト数』

605 ページの『outbound_bytes_sent 送信されたアウトバウンド・バイト数』

677 ページの『prev_uow_stop_time 直前の作業単位完了タイム・スタンプ』

701 ページの『rollback_sql_stmts 試行されたロールバック・ステートメント』

711 ページの『rows_selected 選択行数』

735 ページの『sql_stmts 試行された SQL ステートメントの数』

818 ページの『tpmon_acc_str TP モニター・クライアント・アカウンティング・ストリング : モニター・エレメント』

819 ページの『tpmon_client_app TP モニター・クライアント・アプリケーション名 : モニター・エレメント』

820 ページの『tpmon_client_userid TP モニター・クライアント・ユーザー ID : モニター・エレメント』

820 ページの『tpmon_client_wkstn TP モニター・クライアント・ワークステーション名 : モニター・エレメント』

827 ページの『uow_comp_status 作業単位完了状況』

827 ページの『uow_elapsed_time 最新の作業単位の経過時間』

829 ページの『uow_start_time 作業単位開始タイム・スタンプ』

830 ページの『uow_stop_time 作業単位停止タイム・スタンプ : モニター・エレメント』

844 ページの『xid トランザクション ID』

dcs_appl_info 論理データ・グループ

364 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

366 ページの『agent_status DCS アプリケーション・エージェント』

377 ページの『appl_id - アプリケーション ID :』

381 ページの『appl_name アプリケーション名 : モニター・エレメント』

395 ページの『auth_id 許可 ID』

412 ページの『client_pid クライアント・プロセス ID』

412 ページの『client_platform クライアント・オペレーティング・プラットフォーム』

413 ページの『client_prdid クライアント製品およびバージョン ID : モニター・エレメント』

413 ページの『client_protocol クライアント通信プロトコル』

416 ページの『codepage_id アプリケーションで使用するコード・ページ ID』

457 ページの『dcs_appl_status DCS アプリケーション状況』

457 ページの『dcs_db_name DCS データベース名』

484 ページの『execution_id ユーザー・ログイン ID』

509 ページの『gw_db_alias ゲートウェイでのデータベース別名』

523 ページの『host_ccsid ホスト・コード化文字セット ID』

524 ページの『host_db_name ホスト・データベース名』

524 ページの『host_prdid - ホスト製品/バージョン ID』

526 ページの『inbound_comm_address インバウンド通信アドレス』

603 ページの『outbound_appl_id アウトバウンド・アプリケーション ID』

606 ページの『outbound_comm_address アウトバウンド通信アドレス』

606 ページの『outbound_comm_protocol アウトバウンド通信プロトコル』

607 ページの『outbound_sequence_no アウトバウンド・シーケンス番号』

719 ページの『sequence_no シーケンス番号 : モニター・エレメント』

747 ページの『status_change_time アプリケーション状況変更時刻』

dcs_dbase 論理データ・グループ

417 ページの『commit_sql_stmts 試行されたコミット・ステートメント』

419 ページの『con_elapsed_time 最新の接続経過時間』

419 ページの『con_response_time 接続の最新応答時間』

457 ページの『dcs_db_name DCS データベース名』

478 ページの『elapsed_exec_time ステートメント実行経過時間』

484 ページの『failed_sql_stmts 失敗したステートメント操作』

507 ページの『gw_comm_error_time 通信エラー時刻』

507 ページの『gw_comm_errors 通信エラー』

508 ページの『gw_con_time DB2 Connect ゲートウェイの最初の接続開始』

508 ページの『gw_connections_top ホスト・データベースへの同時接続の最大数』

508 ページの『gw_cons_wait_client クライアントの要求送信を待機している接続の数』

509 ページの『gw_cons_wait_host ホストの応答を待機している接続の数』

509 ページの『gw_cur_cons DB2 Connect の現在の接続数』

510 ページの『gw_total_cons DB2 Connect の接続試行合計回数』

524 ページの『host_db_name ホスト・データベース名』

524 ページの『host_response_time ホスト応答時間』

525 ページの『inbound_bytes_received 受信されたインバウンド・バイト数』

543 ページの『last_reset 最後のリセット・タイム・スタンプ』

573 ページの『max_data_received_1024 受信アウトバウンド・バイト数が 513 から 1024 バイトのステートメント数』

573 ページの『max_data_received_128 受信アウトバウンド・バイト数が 1 から 128 バイトのステートメント数』

574 ページの『max_data_received_16384 受信アウトバウンド・バイト数が 8193 から 16384 バイトのステートメント数』

574 ページの『max_data_received_2048 受信アウトバウンド・バイト数が 1025 から 2048 バイトのステートメント数』

575 ページの『max_data_received_256 受信アウトバウンド・バイト数が 129 から 256 バイトのステートメント数』

575 ページの『max_data_received_31999 受信アウトバウンド・バイト数が 16385 から 31999 バイトのステートメント数 : モニター・エレメント』

576 ページの『max_data_received_4096 受信アウトバウンド・バイト数が 2049 から 4096 バイトのステートメント数』

576 ページの『max_data_received_512 受信アウトバウンド・バイト数が 257 から 512 バイトのステートメント数』

577 ページの『max_data_received_64000 受信アウトバウンド・バイト数が 32000 から 64000 バイトのステートメント数 : モニター・エレメント』

577 ページの『max_data_received_8192 受信アウトバウンド・バイト数が 4097 から 8192 バイトのステートメント数』

577 ページの『max_data_received_gt64000 受信アウトバウンド・バイト数が 64000 バイトを超えるステートメント数』

578 ページの『max_data_sent_1024 送信アウトバウンド・バイト数が 513 から 1024 バイトのステートメント数』

578 ページの『max_data_sent_128 送信アウトバウンド・バイト数が 1 から 128 バイトのステートメント数』

579 ページの『max_data_sent_16384 送信アウトバウンド・バイト数が 8193 から 16384 バイトのステートメント数』

579 ページの『max_data_sent_2048 送信アウトバウンド・バイト数が 1025 から 2048 バイトのステートメント数』

580 ページの『max_data_sent_256 送信アウトバウンド・バイト数が 129 から 256 バイトのステートメント数』

580 ページの『max_data_sent_31999 送信アウトバウンド・バイト数が 16385 から 31999 バイトのステートメント数』

581 ページの『max_data_sent_4096 送信アウトバウンド・バイト数が 2049 から 4096 バイトのステートメント数』

581 ページの『max_data_sent_512 送信アウトバウンド・バイト数が 257 から 512 バイトのステートメント数』

582 ページの『max_data_sent_64000 送信アウトバウンド・バイト数が 32000 から 64000 バイトのステートメント数』

582 ページの『max_data_sent_8192 送信アウトバウンド・バイト数が 4097 から 8192 バイトのステートメント数』

583 ページの『max_data_sent_gt64000 送信アウトバウンド・バイト数が 64000 バイトを超えるステートメント数』

583 ページの『max_network_time_100_ms ネットワーク時間が 16 から 100 ミリ秒のステートメント数』

584 ページの『max_network_time_16_ms ネットワーク時間が 4 から 16 ミリ秒のステートメント数』

584 ページの『max_network_time_1_ms ネットワーク時間が 1 ミリ秒以下のステートメント数』

585 ページの『max_network_time_4_ms ネットワーク時間が 1 から 4 ミリ秒のステートメント数』

585 ページの『max_network_time_500_ms ネットワーク時間が 100 から 500 ミリ秒のステートメント数』

586 ページの『max_network_time_gt500_ms ネットワーク時間が 500 ミリ秒を超えるステートメント数』

588 ページの『network_time_bottom ステートメントの最小ネットワーク時間』

589 ページの『network_time_top ステートメントの最大ネットワーク時間』

605 ページの『outbound_bytes_sent 送信されたアウトバウンド・バイト数』

701 ページの『rollback_sql_stmts 試行されたロールバック・ステートメント』

711 ページの『rows_selected 選択行数』

735 ページの『sql_stmts 試行された SQL ステートメントの数』

dcs_stmt 論理データ・グループ

400 ページの『blocking_cursor ブロック・カーソル』

446 ページの『creator アプリケーション作成者』

478 ページの『elapsed_exec_time ステートメント実行経過時間』

502 ページの『fetch_count 成功したフェッチの数』

510 ページの『gw_exec_time DB2 Connect ゲートウェイ処理の経過時間』

524 ページの『host_response_time ホスト応答時間』

525 ページの『inbound_bytes_received 受信されたインバウンド・バイト数』

526 ページの『inbound_bytes_sent 送信されたインバウンド・バイト数』

599 ページの『num_transmissions 伝送回数』

599 ページの『num_transmissions_group 伝送グループの回数』

604 ページの『outbound_bytes_received 受信されたアウトバウンド・バイト数』

605 ページの『outbound_bytes_sent 送信されたアウトバウンド・バイト数』

608 ページの『package_name - パッケージ名 : モニター・エレメント』

684 ページの『query_card_estimate 照会行数の見積もり』

685 ページの『query_cost_estimate - 照会コストの見積もり : モニター・エレメント』

716 ページの『section_number - セクション番号 : モニター・エレメント』

747 ページの『stmt_elapsed_time 最新のステートメント経過時間』

751 ページの『stmt_operation/operation ステートメント操作 : モニター・エレメント』

755 ページの『stmt_start ステートメント操作開始タイム・スタンプ』

756 ページの『stmt_stop ステートメント操作停止タイム・スタンプ』

757 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

detail_log 論理データ・グループ

447 ページの『current_active_log 現行アクティブ・ログ・ファイル番号』

447 ページの『current_archive_log 現行アーカイブ・ログ・ファイル番号』

503 ページの『first_active_log 先頭アクティブ・ログ・ファイル番号』

542 ページの『last_active_log 最終アクティブ・ログ・ファイル番号』

590 ページの『node_number ノード番号』

dynsql 論理データ・グループ

502 ページの『fetch_count 成功したフェッチの数』

529 ページの『insert_timestamp - ステートメント挿入タイムスタンプ : モニター・エレメント』

534 ページの『int_rows_deleted 削除された内部行数』

534 ページの『int_rows_inserted 挿入された内部行数』

535 ページの『int_rows_updated 更新された内部行数』

592 ページの『num_compilations ステートメント・コンパイル数』

592 ページの『num_executions - ステートメント実行回数 : モニター・エレメント』

632 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

634 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

641 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

643 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

652 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

654 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

655 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

657 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

659 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

661 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

665 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

667 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

676 ページの『prep_time_best ステートメント最短準備時間 : モニター・エレメント』

676 ページの『prep_time_worst ステートメント最長準備時間 : モニター・エレメント』

707 ページの『rows_read - 読み取り行数 : モニター・エレメント』

712 ページの『rows_written 書き込み行数』

731 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

746 ページの『stats_fabricate_time - 統計作成アクティビティーに費やされた合計時間: モニター・エレメント』

753 ページの『stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID』

754 ページの『stmt_sorts ステートメント・ソート回数』

757 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

764 ページの『sync_runstats_time - 同期 RUNSTATS アクティビティーに費やされた合計時間: モニター・エレメント』

805 ページの『total_exec_time ステートメント実行の経過時間』

813 ページの『total_sort_time - ソート時間合計 : モニター・エレメント』

815 ページの『total_sys_cpu_time ステートメントのシステム CPU の合計時間 : モニター・エレメント』

817 ページの『total_usr_cpu_time ステートメントのユーザー CPU の合計時間 : モニター・エレメント』

dynsql_list 論理データ・グループ

452 ページの『db_name データベース名』

453 ページの『db_path データベース・パス』

fcm 論理データ・グループ

403 ページの『buff_free 現在空いている FCM バッファー』

403 ページの『buff_free_bottom 空き FCM バッファーの最小数』

408 ページの『ch_free 現在空いているチャネル』

408 ページの『ch_free_bottom 空いているチャネルの最小』

fcm_node 論理データ・グループ

429 ページの『connection_status 接続状況』

590 ページの『node_number ノード番号』

802 ページの『total_buffers_rcvd 受信された FCM バッファーの合計』

803 ページの『total_buffers_sent 送信された FCM バッファーの合計』

hadr 論理データ・グループ

510 ページの『hadr_connect_status HADR 接続状況 : モニター・エレメント』

511 ページの『hadr_connect_time HADR 接続時刻 : モニター・エレメント』

512 ページの『hadr_heartbeat HADR ハートビート : モニター・エレメント』

512 ページの『hadr_local_host - HADR ローカル・ホスト : モニター・エレメント』

513 ページの『hadr_local_service HADR ローカル・サービス : モニター・エレメント』

513 ページの『hadr_log_gap HADR ログ・ギャップ』

515 ページの『hadr_primary_log_file HADR 1 次ログ・ファイル : モニター・エレメント』

515 ページの『hadr_primary_log_lsn HADR 1 次ログ LSN : モニター・エレメント』

516 ページの『hadr_primary_log_page HADR 1 次ログ・ページ : モニター・エレメント』

516 ページの『hadr_remote_host HADR リモート・ホスト : モニター・エレメント』

516 ページの『hadr_remote_instance HADR リモート・インスタンス : モニター・エレメント』

517 ページの『hadr_remote_service HADR リモート・サービス : モニター・エレメント』

517 ページの『hadr_role HADR の役割』

518 ページの『hadr_standby_log_file HADR スタンバイ・ログ・ファイル : モニター・エレメント』

518 ページの『hadr_standby_log_lsn HADR スタンバイ・ログ LSN : モニター・エレメント』

519 ページの『hadr_standby_log_page HADR スタンバイ・ログ・ページ : モニター・エレメント』

519 ページの『hadr_state HADR の状態 : モニター・エレメント』

520 ページの『hadr_syncmode HADR 同期モード : モニター・エレメント』

520 ページの『hadr_timeout HADR タイムアウト : モニター・エレメント』

lock 論理データ・グループ

449 ページの『data_partition_id - データ・パーティション ID : モニター・エレメント』

546 ページの『lock_attributes ロック属性 : モニター・エレメント』

547 ページの『lock_count ロック・カウント : モニター・エレメント』

548 ページの『lock_current_mode 変換前の元のロック・モード』

548 ページの『lock_escalation ロック・エスカレーション : モニター・エレメント』

551 ページの『lock_hold_count ロック保留カウント : モニター・エレメント』

552 ページの『lock_mode - ロック・モード : モニター・エレメント』

553 ページの『lock_name ロック名 : モニター・エレメント』

554 ページの『lock_object_name ロック対象名』

555 ページの『lock_object_type - 待機中のロック対象タイプ : モニター・エレメント』

556 ページの『lock_release_flags ロック保留解除フラグ : モニター・エレメント』

557 ページの『lock_status - ロック状況 : モニター・エレメント』

590 ページの『node_number ノード番号』

766 ページの『table_file_id - 表ファイル ID : モニター・エレメント』

766 ページの『table_name - 表名 : モニター・エレメント』

768 ページの『table_schema - 表スキーマ名 : モニター・エレメント』

776 ページの『tablespace_name - 表スペース名 : モニター・エレメント』

lock_wait 論理データ・グループ

365 ページの『agent_id_holding_lock ロックを保持しているエージェント ID』

379 ページの『appl_id_holding_lk ロックを保持しているアプリケーション ID』

449 ページの『data_partition_id - データ・パーティション ID : モニター・エレメント』

546 ページの『lock_attributes ロック属性 : モニター・エレメント』

548 ページの『lock_current_mode 変換前の元のロック・モード』

548 ページの『lock_escalation ロック・エスカレーション : モニター・エレメント』

552 ページの『lock_mode - ロック・モード : モニター・エレメント』

553 ページの『lock_mode_requested 要求されているロック・モード : モニター・エレメント』

553 ページの『lock_name ロック名 : モニター・エレメント』

555 ページの『lock_object_type - 待機中のロック対象タイプ : モニター・エレメント』

556 ページの『lock_release_flags ロック保留解除フラグ : モニター・エレメント』

559 ページの『lock_wait_start_time ロック待機開始タイム・スタンプ』

590 ページの『node_number ノード番号』

742 ページの『ss_number サブセクション番号』

766 ページの『table_name - 表名 : モニター・エレメント』

768 ページの『table_schema - 表スキーマ名 : モニター・エレメント』

776 ページの『tablespace_name - 表スペース名 : モニター・エレメント』

memory_pool 論理データ・グループ

590 ページの『node_number ノード番号』

630 ページの『pool_config_size メモリー・プールの構成済みサイズ』

631 ページの『pool_cur_size メモリー・プールの現行サイズ』

640 ページの『pool_id メモリー・プール ID』

651 ページの『pool_secondary_id メモリー・プール 2 次 ID』

662 ページの『pool_watermark メモリー・プール水準点』

progress 論理データ・グループ

680 ページの『progress_completed_units 完了した進行作業単位』

681 ページの『progress_description 進行の記述』

682 ページの『progress_seq_num 進行シーケンス番号』

682 ページの『progress_start_time 進行開始時刻』

683 ページの『progress_total_units 合計進行作業単位』

683 ページの『progress_work_metric 進行作業メトリック』

progress_list 論理データ・グループ

681 ページの『progress_list_attr 現在の進行リストの属性』

682 ページの『progress_list_cur_seq_num 現行の進行リストのシーケンス番号』

rollforward 論理データ・グループ

590 ページの『node_number ノード番号』

700 ページの『rf_log_num ロールフォワードされているログ』

700 ページの『rf_status ログ・フェーズ』

701 ページの『rf_timestamp ロールフォワード・タイム・スタンプ』

701 ページの『rf_type ロールフォワード・タイプ』

825 ページの『ts_name - ロールフォワードされている表スペース : モニター・エレメント』

stmt 論理データ・グループ

685 ページの『query_cost_estimate - 照会コストの見積もり : モニター・エレメント』

707 ページの『rows_read - 読み取り行数 : モニター・エレメント』

712 ページの『rows_written 書き込み行数』

716 ページの『section_number - セクション番号 : モニター・エレメント』

731 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

747 ページの『stmt_elapsed_time 最新のステートメント経過時間』

751 ページの『stmt_node_number ステートメント・ノード』

751 ページの『stmt_operation/operation ステートメント操作 : モニター・エレメント』

754 ページの『stmt_sorts ステートメント・ソート回数』

755 ページの『stmt_start ステートメント操作開始タイム・スタンプ』

756 ページの『stmt_stop ステートメント操作停止タイム・スタンプ』

756 ページの『stmt_sys_cpu_time ステートメントが使用したシステム CPU 時間』

757 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

758 ページの『stmt_type ステートメント・タイプ : モニター・エレメント』

759 ページの『stmt_usr_cpu_time ステートメントに使用されたユーザー CPU 時間』

813 ページの『total_sort_time - ソート時間合計 : モニター・エレメント』

stmt_transmissions 論理データ・グループ

478 ページの『elapsed_exec_time ステートメント実行経過時間』

524 ページの『host_response_time ホスト応答時間』

573 ページの『max_data_received_1024 受信アウトバウンド・バイト数が 513 から 1024 バイトのステートメント数』

573 ページの『max_data_received_128 受信アウトバウンド・バイト数が 1 から 128 バイトのステートメント数』

574 ページの『max_data_received_16384 受信アウトバウンド・バイト数が 8193 から 16384 バイトのステートメント数』

574 ページの『max_data_received_2048 受信アウトバウンド・バイト数が 1025 から 2048 バイトのステートメント数』

575 ページの『max_data_received_256 受信アウトバウンド・バイト数が 129 から 256 バイトのステートメント数』

575 ページの『max_data_received_31999 受信アウトバウンド・バイト数が 16385 から 31999 バイトのステートメント数 : モニター・エレメント』

576 ページの『max_data_received_4096 受信アウトバウンド・バイト数が 2049 から 4096 バイトのステートメント数』

576 ページの『max_data_received_512 受信アウトバウンド・バイト数が 257 から 512 バイトのステートメント数』

604 ページの『outbound_bytes_received_bottom 受信された最小アウトバウンド・バイト数』

605 ページの『outbound_bytes_received_top 受信された最大アウトバウンド・バイト数』

605 ページの『outbound_bytes_sent 送信されたアウトバウンド・バイト数』

605 ページの『outbound_bytes_sent_bottom 送信された最小アウトバウンド・バイト数』

606 ページの『outbound_bytes_sent_top 送信された最大アウトバウンド・バイト数』

734 ページの『sql_chains 試行された SQL チェーンの数』

735 ページの『sql_stmts 試行された SQL ステートメントの数』

subsection 論理データ・グループ

707 ページの『rows_read - 読み取り行数 : モニター・エレメント』

712 ページの『rows_written 書き込み行数』

741 ページの『ss_exec_time サブセクション実行経過時間』

741 ページの『ss_node_number サブセクション・ノード番号』

742 ページの『ss_number サブセクション番号』

742 ページの『ss_status サブセクションの状況』

742 ページの『ss_sys_cpu_time サブセクションに使用されたシステム CPU 時間』

743 ページの『ss_usr_cpu_time サブセクションに使用されたユーザー CPU 時間』

821 ページの『tq_cur_send_spills オーバーフローした表キュー・バッファーの現在数 : モニター・エレメント』

821 ページの『tq_id_waiting_on ノード上の表キュー待機』

822 ページの『tq_max_send_spills 表キュー・バッファー・オーバーフローの最大数』

822 ページの『tq_node_waited_for 表キュー上のノード待機』

823 ページの『tq_rows_read 表キューから読み取られた行数』

823 ページの『tq_rows_written 表キューに書き込まれた行数』

824 ページの『tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数 : モニター・エレメント』

825 ページの『tq_wait_for_any 表キュー上のノード送信待機』

table 論理データ・グループ

449 ページの『data_object_pages データ・オブジェクト・ページ数』

449 ページの『data_partition_id - データ・パーティション ID : モニター・エレメント』

527 ページの『index_object_pages 索引オブジェクト・ページ数』

544 ページの『lob_object_pages LOB オブジェクト・ページ数』

572 ページの『long_object_pages 長いオブジェクト・ページ数』

607 ページの『overflow_accesses - オーバーフロー・レコードへのアクセス : モニター・エレメント』

610 ページの『page_reorgs ページ再編成』

707 ページの『rows_read - 読み取り行数 : モニター・エレメント』

712 ページの『rows_written 書き込み行数』

766 ページの『table_file_id - 表ファイル ID : モニター・エレメント』

766 ページの『table_name - 表名 : モニター・エレメント』

768 ページの『table_schema - 表スキーマ名 : モニター・エレメント』

769 ページの『table_type - 表タイプ : モニター・エレメント』

773 ページの『tablespace_id - 表スペース ID : モニター・エレメント』

844 ページの『xda_object_pages XDA オブジェクト・ページ数』

table_list 論理データ・グループ

451 ページの『db_conn_time データベース活動化タイム・スタンプ : モニター・エレメント』

452 ページの『db_name データベース名』

453 ページの『db_path データベース・パス』

528 ページの『input_db_alias 入力データベース別名』

543 ページの『last_reset 最後のリセット・タイム・スタンプ』

table_reorg 論理データ・グループ

449 ページの『data_partition_id - データ・パーティション ID : モニター・エレメント』

693 ページの『reorg_completion 再編成完了フラグ』

694 ページの『reorg_current_counter 再編成の進行状況』

694 ページの『reorg_end 表再編成終了時刻』

694 ページの『reorg_index_id 表の再編成に使用される索引』

695 ページの『reorg_max_counter 再編成の合計量』

695 ページの『reorg_max_phase 再編成の最大フェーズ数』

695 ページの『reorg_phase - 表の再編成のフェーズ : モニター・エレメント』

696 ページの『reorg_phase_start 表再編成フェーズ開始時刻』

696 ページの『reorg_rows_compressed - 圧縮行数』

697 ページの『reorg_rows_rejected_for_compression - 圧縮がリジェクトされる行』

697 ページの『reorg_start 表再編成開始時刻』

697 ページの『reorg_status 表再編成の状況』

698 ページの『reorg_tbspc_id - 表またはデータ・パーティションが再編成される表スペース』

698 ページの『reorg_type 表再編成の属性』

699 ページの『reorg_xml_regions_compressed – 圧縮された XML 領域 : モニター・エレメント』

699 ページの『reorg_xml_regions_rejected_for_compression – 圧縮を拒否された XML 領域 : モニター・エレメント』

tablespace 論理データ・グループ

465 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

467 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

468 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

470 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

471 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

473 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

502 ページの『files_closed - クローズしたデータベース・ファイル : モニター・エレメント』

504 ページの『fs_caching - ファイル・システム・キャッシング : モニター・エレメント』

621 ページの『pool_async_data_read_reqs - バッファー・プール非同期読み取り要求 : モニター・エレメント』

622 ページの『pool_async_data_reads バッファー・プール非同期データ読み取り : モニター・エレメント』

623 ページの『pool_async_data_writes - バッファー・プール非同期データ書き込み : モニター・エレメント』

624 ページの『pool_async_index_read_reqs - バッファー・プール非同期索引読み取り要求 : モニター・エレメント』

624 ページの『pool_async_index_reads - バッファー・プール非同期索引読み取り : モニター・エレメント』

625 ページの『pool_async_index_writes - バッファー・プール非同期索引書き込み : モニター・エレメント』

626 ページの『pool_async_read_time バッファー・プール非同期読み取り時間』

627 ページの『pool_async_write_time バッファー・プール非同期書き込み時間』

628 ページの『pool_async_xda_read_reqs - バッファー・プール非同期 XDA 読み取り要求 : モニター・エレメント』

629 ページの『pool_async_xda_reads - バッファー・プール非同期 XDA データ読み取り : モニター・エレメント』

630 ページの『pool_async_xda_writes - バッファー・プール非同期 XDA データ書き込み : モニター・エレメント』

632 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

634 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

635 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

641 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

643 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

645 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

648 ページの『pool_no_victim_buffer - バッファー・プールの非ビクティム・バッファー数 : モニター・エレメント』

649 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

652 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

654 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

655 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

657 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

659 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

661 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

663 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

665 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

667 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

669 ページの『pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント』

770 ページの『tablespace_auto_resize_enabled - 表スペースの自動サイズ変更可能 : モニター・エレメント』

771 ページの『tablespace_content_type - 表スペースのコンテンツ・タイプ : モニター・エレメント』

771 ページの『tablespace_cur_pool_id - 現在使用中のバッファー・プール : モニター・エレメント』

772 ページの『tablespace_extent_size - 表スペースのエクステント・サイズ : モニター・エレメント』

773 ページの『tablespace_id - 表スペース ID : モニター・エレメント』

776 ページの『tablespace_name - 表スペース名 : モニター・エレメント』

777 ページの『tablespace_next_pool_id - 次の始動時に使用されるバッファー・プール : モニター・エレメント』

778 ページの『tablespace_page_size - 表スペースのページ・サイズ : モニター・エレメント』

780 ページの『tablespace_prefetch_size - 表スペースのプリフェッチ・サイズ : モニター・エレメント』

782 ページの『tablespace_rebalancer_mode - リバランサー・モード : モニター・エレメント』

787 ページの『tablespace_type - 表スペース・タイプ : モニター・エレメント』

788 ページの『tablespace_using_auto_storage - 自動ストレージが使用可能な表スペース : モニター・エレメント』

tablespace_container 論理データ・グループ

430 ページの『container_accessible - コンテナーのアクセス可能性 : モニター・エレメント』

431 ページの『container_id - コンテナー ID : モニター・エレメント』

431 ページの『container_name - コンテナー名 : モニター・エレメント』

431 ページの『container_stripe_set - コンテナー・ストライプ・セット : モニター・エレメント』

432 ページの『container_total_pages - コンテナー内の合計ページ数 : モニター・エレメント』

432 ページの『container_type - コンテナー・タイプ : モニター・エレメント』

433 ページの『container_usable_pages - コンテナー内の使用可能なページ数 : モニター・エレメント』

tablespace_list 論理データ・グループ

451 ページの『db_conn_time データベース活動化タイム・スタンプ : モニター・エレメント』

452 ページの『db_name データベース名』

453 ページの『db_path データベース・パス』

528 ページの『input_db_alias 入力データベース別名』

543 ページの『last_reset 最後のリセット・タイム・スタンプ』

tablespace_nodeinfo 論理データ・グループ

772 ページの『tablespace_current_size 表スペースの現行サイズ』

772 ページの『tablespace_free_pages 表スペース内のフリー・ページ数 : モニター・エレメント』

773 ページの『tablespace_increase_size バイト単位のサイズの増加』

774 ページの『tablespace_increase_size_percent パーセント単位のサイズの増加 : モニター・エレメント』

774 ページの『tablespace_initial_size 表スペースの初期サイズ』

774 ページの『tablespace_last_resize_failed 失敗した最後のサイズ変更』

775 ページの『tablespace_last_resize_time 最後にサイズ変更が正常に行われた時刻』

775 ページの『tablespace_max_size 表スペースの最大サイズ』

tablespace_quiescer 論理データ・グループ

tablespace_range 論理データ・グループ

689 ページの『range_max_extent 範囲内の最大エクステント』
689 ページの『range_max_page_number 範囲内の最大ページ』
690 ページの『range_num_containers 範囲内コンテナーの数』
690 ページの『range_number 範囲番号』
690 ページの『range_offset 範囲オフセット』
690 ページの『range_start_stripe 開始ストライプ』
690 ページの『range_stripe_set_number ストライプ・セット番号』

utility_info 論理データ・グループ

590 ページの『node_number ノード番号』
833 ページの『utility_dbname ユーティリティーで操作されるデータベース』
833 ページの『utility_description ユーティリティー記述』
833 ページの『utility_id ユーティリティー ID』
834 ページの『utility_invoker_type - ユーティリティー呼び出し側タイプ』
834 ページの『utility_priority ユーティリティー優先度』
834 ページの『utility_start_time ユーティリティー開始時刻』
834 ページの『utility_state - ユーティリティー状態』
835 ページの『utility_type ユーティリティー・タイプ』

イベント・タイプの論理データ・グループへのマッピング

ファイル、パイプおよび表のイベント・モニターに関しては、イベント・モニターの出力は、順序付けされた一連の論理データ・グループから成っています。どのイベント・モニター・タイプの場合にも、出力レコードには必ず同じ開始論理データ・グループが含まれています。論理データ・グループが示すフレームは、イベント・モニターによって記録されるイベント・タイプによって異なります。

ファイルおよびパイプ・イベント・モニターの場合、イベント・レコードはどの接続についても生成されることがあるため、ストリーム中にいろいろな順序で現れる場合があります。すなわち、接続 1 のトランザクション・イベントの直後に接続 2 の接続イベントを取得することがあるということです。しかし、単一の接続または単一のイベントに属するレコードは、論理順序で現れます。例えば、ステートメント・レコード (ステートメントの終わり) は、トランザクション・レコード (UOW の終わり) があれば、必ずその前に来ます。同様に、デッドロック・イベント・レコードは、必ずデッドロックに関係する各接続のデッドロック接続イベント・レコードの前に来ます。**アプリケーション ID** または**アプリケーション・ハンドル (agent_id)** を使って、レコードと接続を一致させることができます。

接続ヘッダー・イベントは通常、データベースへのそれぞれの接続ごとに書き込まれます。詳細付きデッドロック・イベント・モニターの場合は、デッドロックが生じたときにだけ書き込まれます。この場合、接続ヘッダー・イベントは、デッドロックに関係したものについてのみ書き込まれます。データベースへのすべての接続について書き込まれるわけではありません。

論理データ・グループは、4 つの異なるレベルに従って配列されます。すなわちモニター、プロローグ、内容、およびエピローグです。以下は、各レベルの詳細な記述です。対応するイベント・タイプおよび論理データ・グループも示しています。

モニター

モニター・レベルの情報は、すべてのイベント・モニターに対して生成されます。これは、イベント・モニターのメタデータから成っています。

表 72. イベント・モニター・データ・ストリーム : モニター・セクション

イベント・タイプ	論理データ・グループ	入手できる情報
モニター・レベル	event_log_stream_header	イベント・モニターのバージョン・レベルおよびバイトの並び順を識別する。アプリケーションはこのヘッダーを使用して、 evmon 出力ストリームを処理できるかどうかを判別できます。

プロローグ

プロローグ情報は、イベント・モニターが活動化されると生成されます。

表 73. イベント・モニター・データ・ストリーム : プロローグ・セクション

イベント・タイプ	論理データ・グループ	入手できる情報
ログ・ヘッダー	event_log_header	トレースの特性、例えば、サーバーのタイプおよびメモリーのレイアウト。
データベース・ヘッダー	event_db_header	データベース名、パスおよび活動化時間。
イベント・モニター開始	event_start	モニターが開始されるか再始動された時間。
接続ヘッダー	event_connheader	現行接続のヘッダーごとに 1 つ。接続時間とアプリケーション名を含みます。イベント接続ヘッダーが生成されるのは、接続、ステートメント、トランザクション、およびデッドロックの各イベント・モニターに対してのみです。詳細付きデッドロック・イベント・モニターは、デッドロックが生じたときだけ接続ヘッダーを生成します。

内容

イベント・モニターの指定されたイベント・タイプに固有の情報は、内容セクションに表示されます。

表 74. イベント・モニター・データ・ストリーム : 内容セクション

イベント・タイプ	論理データ・グループ	入手できる情報
ステートメント・イベント	event_stmt	ステートメント・レベル・データ。動的ステートメントのテキストを含む。ステートメント・イベント・モニターは、フェッチのログを取りません。
サブセクション・イベント	event_subsection	サブセクション・レベル・データ。
トランザクション・イベント[1]	event_xact	トランザクション・レベル・データ。
接続イベント	event_conn	接続レベル・データ。
デッドロック・イベント	event_deadlock	デッドロック・レベル・データ。
デッドロック接続イベント	event_dlconn	デッドロックに関係している接続ごとに 1 つ。関係するアプリケーションと競合しているロックを含みます。
詳細付きデッドロック接続イベント	event_detailed_dlconn、lock	デッドロックに関係している接続ごとに 1 つ。関係するアプリケーション、競合しているロック、現在のステートメント情報、およびアプリケーション競合によって保持された他のロックを含みます。
オーバーフロー	event_overflow	脱落したレコードの数。書き込み装置が (ブロック化されていない) イベント・モニターに追い付かないときに生成されます。
詳細付きデッドロック履歴[2]	event_stmt_history	デッドロックに関係する作業単位で実行されたステートメントのリスト。
詳細付きデッドロック履歴の値[2]	event_data_value	event_stmt_history リスト内のステートメント用のパラメーター・マーカー。
アクティビティー	event_activity	システムで実行が完了した、または完了前にキャプチャーされたアクティビティーのリスト。
	event_activitystmt	アクティビティー・タイプがステートメントであった際に、そのアクティビティーが実行したステートメントに関する情報。
	event_activityvals	SQL ステートメントである各アクティビティーの入力変数として使用されるデータ値。こうしたデータ値には、LOB データ、LONG データ、または構造化タイプ・データは含まれません。

表 74. イベント・モニター・データ・ストリーム : 内容セクション (続き)

イベント・タイプ	論理データ・グループ	入手できる情報
統計	event_scstats	システム内のそれぞれのサービス・クラス、処理クラス、またはワークロードで実行されたアクティビティーから算出される統計、さらにしきい値キューから算出される統計。
	event_wcstats	
	event_wlstats	
	event_qstats	
	event_histogrambin	
しきい値違反	event_threshold_violations	違反したしきい値とその時間を示す情報。

1 このオプションは推奨されなくなりました。 このオプションの使用は推奨されておらず、将来のリリースでは除去される予定です。 CREATE EVENT MONITOR FOR UNIT OF WORK ステートメントを使用して、トランザクション・イベントをモニターしてください。

2 このオプションは推奨されなくなりました。 このオプションの使用は推奨されておらず、将来のリリースでは除去される予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

エピローグ

エピローグ情報は、データベースが非活動化状態にあるとき (最後のアプリケーションが切断を終了したとき) に生成されます。

表 75. イベント・モニター・データ・ストリーム : エピローグ・セクション

イベント・タイプ	論理データ・グループ	入手できる情報
データベース・イベント	event_db	データベース・マネージャー・レベル・データ。
バッファー・プール・イベント	event_bufferpool	バッファー・プール・レベル・データ。
表スペース・イベント	event_tablespace	表スペース・レベル・データ。
表イベント	event_table	表レベル・データ。

イベント・モニターの論理データ・グループおよびモニター・エレメント

次の表は、論理データ・グループと、イベント・モニターによって戻されるモニター・エレメントの一覧表です。

- 322 ページの『event_activity 論理データ・グループ』
- 324 ページの『event_activitystmt 論理データ・グループ』
- 325 ページの『event_activityvals 論理データ・グループ』
- 326 ページの『event_bufferpool 論理データ・グループ』
- 327 ページの『event_conn 論理データ・グループ』
- 330 ページの『event_connheader 論理データ・グループ』

- 330 ページの『event_connmemuse 論理データ・グループ』
- 330 ページの『event_data_value 論理データ・グループ』
- 331 ページの『event_db 論理データ・グループ』
- 335 ページの『event_dbheader 論理データ・グループ』
- 335 ページの『event_dbmemuse 論理データ・グループ』
- 335 ページの『event_deadlock 論理データ・グループ』
- 335 ページの『event_detailed_dlconn 論理データ・グループ』
- 336 ページの『event_dlconn 論理データ・グループ』
- 337 ページの『event_histogrambin 論理データ・グループ』
- 338 ページの『event_log_header 論理データ・グループ』
- 338 ページの『event_overflow 論理データ・グループ』
- 338 ページの『event_qstats 論理データ・グループ』
- 339 ページの『event_scstats 論理データ・グループ』
- 340 ページの『event_start 論理データ・グループ』
- 340 ページの『event_stmt 論理データ・グループ』
- 341 ページの『event_stmt_history 論理データ・グループ』
- 342 ページの『event_subsection 論理データ・グループ』
- 342 ページの『event_table 論理データ・グループ』
- 343 ページの『event_tablespace 論理データ・グループ』
- 344 ページの『event_thresholdviolations 論理データ・グループ』
- 345 ページの『event_wlstats 論理データ・グループ』
- 346 ページの『event_wcstats 論理データ・グループ』
- 347 ページの『event_xact 論理データ・グループ』
- 348 ページの『lock 論理データ・グループ』
- 348 ページの『sqlca 論理データ・グループ』

event_activity 論理データ・グループ

355 ページの『act_exec_time アクティビティー実行時間 : モニター・エレメント』

358 ページの『activate_timestamp タイム・スタンプの活動化 : モニター・エレメント』

360 ページの『activity_id アクティビティー ID : モニター・エレメント』

360 ページの『activity_secondary_id アクティビティー 2 次 ID : モニター・エレメント』

362 ページの『activity_type アクティビティー・タイプ : モニター・エレメント』

363 ページの『address - 接続の開始元となった IP アドレス』

364 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

377 ページの『appl_id - アプリケーション ID :』

381 ページの『appl_name アプリケーション名 : モニター・エレメント』

388 ページの『arm_correlator アプリケーション応答測定相関関係子 : モニター・エレメント』

441 ページの『coord_partition_num コーディネーター・パーティション番号 : モニター・エレメント』

456 ページの『db_work_action_set_id データベース作業アクション・セット ID : モニター・エレメント』

456 ページの『db_work_class_id データベース作業クラス ID : モニター・エレメント』

details_xml (この XML 文書には、MON_GET_ACTIVITY_DETAILS 表関数の出力の DETAILS 列にレポートされるモニター・エレメントがすべて含まれています。)

613 ページの『parent_activity_id 親アクティビティー ID : モニター・エレメント』

613 ページの『parent_uow_id 親作業単位 ID : モニター・エレメント』

614 ページの『partial_record 部分レコード : モニター・エレメント』

632 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

634 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

598 ページの『num_remaps 再マップ数 : モニター・エレメント』

714 ページの『sc_work_action_set_id サービス・クラス作業アクション・セット ID : モニター・エレメント』

715 ページの『sc_work_class_id サービス・クラス作業クラス ID : モニター・エレメント』

724 ページの『service_subclass_name サービス・サブクラス名 : モニター・エレメント』

725 ページの『service_superclass_name サービス・スーパークラス名 : モニター・エレメント』

726 ページの『session_auth_id セッション許可 ID : モニター・エレメント』

731 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

736 ページの『sqlca SQL 連絡域 (SQLCA)』

798 ページの『time_completed 完了時刻 : モニター・エレメント』

798 ページの『time_created 作成時刻 : モニター・エレメント』

799 ページの『time_started 開始時刻 : モニター・エレメント』

813 ページの『total_sort_time - ソート時間合計 : モニター・エレメント』

814 ページの『total_sorts - ソート合計 : モニター・エレメント』

818 ページの『tpmon_acc_str TP モニター・クライアント・アカウンティング・ストリング : モニター・エレメント』

819 ページの『tpmon_client_app TP モニター・クライアント・アプリケーション名 : モニター・エレメント』

820 ページの『tpmon_client_userid TP モニター・クライアント・ユーザー ID : モニター・エレメント』

820 ページの『tpmon_client_wkstn TP モニター・クライアント・ワークステーション名 : モニター・エレメント』

828 ページの『uow_id 作業単位 ID : モニター・エレメント』

840 ページの『workload_id ワークロード ID : モニター・エレメント』

842 ページの『workload_occurrence_id ワークロード・オカレンス ID : モニター・エレメント』

641 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

643 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

652 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

654 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

655 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

657 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

659 ページの『pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント』

661 ページの『pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント』

665 ページの『pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント』

667 ページの『pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント』

676 ページの『prep_time 準備時間 : モニター・エレメント』

684 ページの『query_card_estimate 照会行数の見積もり』

685 ページの『query_cost_estimate - 照会コストの見積もり : モニター・エレメント』

705 ページの『rows_fetched フェッチ行数 : モニター・エレメント』

706 ページの『rows_modified 変更行数 : モニター・エレメント』

709 ページの『rows_returned 戻り行数 : モニター・エレメント』

765 ページの『system_cpu_time システム CPU 時間』

832 ページの『user_cpu_time ユーザー CPU 時間』

event_activitystmt 論理データ・グループ

358 ページの『activate_timestamp タイム・スタンプの活動化 : モニター・エレメント』

360 ページの『activity_id アクティビティー ID : モニター・エレメント』

360 ページの『activity_secondary_id アクティビティー 2 次 ID : モニター・エレメント』

377 ページの『appl_id - アプリケーション ID :』

418 ページの『comp_env_desc コンパイル環境 : モニター・エレメント』

446 ページの『creator アプリケーション作成者』

477 ページの『eff_stmt_text - 有効なステートメント・テキスト : モニター・エレメント』

483 ページの『executable_id 実行可能 ID : モニター・エレメント』

608 ページの『package_name - パッケージ名 : モニター・エレメント』

609 ページの『package_version_id - パッケージ・バージョン : モニター・エレメント』

716 ページの『section_env セクション環境 : モニター・エレメント』

716 ページの『section_number - セクション番号 : モニター・エレメント』

748 ページの『stmt_first_use_time ステートメントの最初の使用時刻』

749 ページの『stmt_isolation ステートメント分離』

750 ページの『stmt_last_use_time ステートメント最終使用時刻 : モニター・エレメント』

750 ページの『stmt_lock_timeout ステートメント・ロック・タイムアウト : モニター・エレメント』

751 ページの『stmt_nest_level ステートメント・ネスト・レベル : モニター・エレメント』

753 ページの『stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID』

754 ページの『stmt_query_id ステートメント照会 ID : モニター・エレメント』

755 ページの『stmt_source_id ステートメント・ソース ID』

757 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

758 ページの『stmt_type ステートメント・タイプ : モニター・エレメント』

828 ページの『uow_id 作業単位 ID : モニター・エレメント』

event_activityvals 論理データ・グループ

358 ページの『activate_timestamp タイム・スタンプの活動化 : モニター・エレメント』

360 ページの『activity_id アクティビティー ID : モニター・エレメント』

360 ページの『activity_secondary_id アクティビティー 2 次 ID : モニター・エレメント』

377 ページの『appl_id - アプリケーション ID :』

759 ページの『stmt_value_data 値データ』

760 ページの『stmt_value_index 値索引』

760 ページの『stmt_value_isnull NULL 値の値 : モニター・エレメント』

761 ページの『stmt_value_isreopt ステートメント再最適化に使用される変数 : モニター・エレメント』

761 ページの『stmt_value_type 値タイプ : モニター・エレメント』

828 ページの『uow_id 作業単位 ID : モニター・エレメント』

event_bufferpool 論理データ・グループ

401 ページの『bp_id バッファー・プール ID : モニター・エレメント』

402 ページの『bp_name - バッファー・プール名 : モニター・エレメント』

452 ページの『db_name データベース名』

453 ページの『db_path データベース・パス』

465 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

467 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

468 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

470 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

471 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

473 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

481 ページの『event_time イベント時刻』

482 ページの『evmon_activates イベント・モニター活動化回数』

483 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

502 ページの『files_closed - クローズしたデータベース・ファイル : モニター・エレメント』

614 ページの『partial_record 部分レコード : モニター・エレメント』

621 ページの『pool_async_data_read_reqs - バッファー・プール非同期読み取り要求 : モニター・エレメント』

622 ページの『pool_async_data_reads バッファー・プール非同期データ読み取り : モニター・エレメント』

623 ページの『pool_async_data_writes - バッファー・プール非同期データ書き込み : モニター・エレメント』

624 ページの『pool_async_index_reads - バッファー・プール非同期索引読み取り : モニター・エレメント』

625 ページの『pool_async_index_writes - バッファー・プール非同期索引書き込み : モニター・エレメント』

626 ページの『pool_async_read_time バッファー・プール非同期読み取り時間』

627 ページの『pool_async_write_time バッファー・プール非同期書き込み時間』

632 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

634 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

635 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

641 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

643 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

645 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

649 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

663 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

event_conn 論理データ・グループ

352 ページの『acc_curs_blk 受け入れられたブロック・カーソル要求』

364 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

377 ページの『appl_id - アプリケーション ID :』

382 ページの『appl_priority アプリケーション・エージェント優先順位』

382 ページの『appl_priority_type アプリケーション優先順位タイプ』

383 ページの『appl_section_inserts セクション挿入数 : モニター・エレメント』

383 ページの『appl_section_lookups - セクション検索』

395 ページの『authority_bitmap ユーザー許可レベル : モニター・エレメント』

396 ページの『authority_lvl ユーザー許可レベル : モニター・エレメント』

398 ページの『binds_precompiles 試行されたバインド/プリコンパイル』

404 ページの『cat_cache_inserts - カタログ・キャッシュ挿入数』

405 ページの『cat_cache_lookups カタログ・キャッシュ検索』

406 ページの『cat_cache_overflows カタログ・キャッシュ・オーバーフロー数』

417 ページの『commit_sql_stmts 試行されたコミット・ステートメント』

457 ページの『ddl_sql_stmts データ定義言語 (DDL) SQL ステートメント』

459 ページの『deadlocks - デッドロック検出数 : モニター・エレメント』

465 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

467 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

468 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

470 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

471 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

473 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

475 ページの『disconn_time データベース非活動化タイム・スタンプ』

476 ページの『dynamic_sql_stmts 試行された動的 SQL ステートメント』

484 ページの『failed_sql_stmts 失敗したステートメント操作』

521 ページの『hash_join_overflows ハッシュ結合のオーバーフロー』

521 ページの『hash_join_small_overflows ハッシュ結合の短精度オーバーフロー』

530 ページの『int_auto_rebinds 内部自動再バインド』

531 ページの『int_commits 内部コミット数』

532 ページの『int_deadlock_rollbacks デッドロックによる内部ロールバック』

532 ページの『int_rollbacks 内部ロールバック数』

534 ページの『int_rows_deleted 削除された内部行数』

534 ページの『int_rows_inserted 挿入された内部行数』

535 ページの『int_rows_updated 更新された内部行数』

548 ページの『lock_escalation ロック・エスカレーション : モニター・エレメント』

558 ページの『lock_timeouts - ロック・タイムアウト数 : モニター・エレメント』

560 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

562 ページの『lock_waits - ロック待機数 : モニター・エレメント』

600 ページの『olap_func_overflows OLAP 関数のオーバーフロー : モニター・エレメント』

614 ページの『partial_record 部分レコード : モニター・エレメント』

535 ページの『int_rows_updated 更新された内部行数』

618 ページの『pkg_cache_inserts パッケージ・キャッシュ挿入』

618 ページの『pkg_cache_lookups パッケージ・キャッシュ検索』

632 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

634 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

635 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

641 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

643 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

645 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

649 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

652 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

654 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

655 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

657 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

663 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

675 ページの『prefetch_wait_time - プリフェッチ待ち時間 : モニター・エレメント』

677 ページの『priv_workspace_num_overflows 専用ワークスペースのオーバーフロー回数』

678 ページの『priv_workspace_section_inserts 専用ワークスペース・セクション挿入』

679 ページの『priv_workspace_section_lookups 専用ワークスペース・セクション検索』

679 ページの『priv_workspace_size_top 専用ワークスペースの最大サイズ』

691 ページの『rej_curs_blk リジェクトされたブロック・カーソル要求』

701 ページの『rollback_sql_stmts 試行されたロールバック・ステートメント』

707 ページの『rows_read - 読み取り行数 : モニター・エレメント』

711 ページの『rows_selected 選択行数』

712 ページの『rows_written 書き込み行数』

718 ページの『select_sql_stmts 実行された選択 SQL ステートメント』

719 ページの『sequence_no シーケンス番号 : モニター・エレメント』

726 ページの『shr_workspace_num_overflows 共有ワークスペースのオーバーフロー回数』

727 ページの『shr_workspace_section_inserts 共有ワークスペース・セクション挿入数』

728 ページの『shr_workspace_section_lookups 共有ワークスペース・セクション検索』

728 ページの『shr_workspace_size_top 最大共有ワークスペース・サイズ』

731 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

744 ページの『static_sql_stmts 試行された静的 SQL ステートメント』

765 ページの『system_cpu_time システム CPU 時間』

805 ページの『total_hash_joins ハッシュ結合の合計』

806 ページの『total_hash_loops ハッシュ・ループの合計』

807 ページの『total_olap_funcs OLAP 関数の合計数 : モニター・エレメント』

809 ページの『total_sec_cons 2 次接続』

813 ページの『total_sort_time - ソート時間合計 : モニター・エレメント』

814 ページの『total_sorts - ソート合計 : モニター・エレメント』

825 ページの『uid_sql_stmts 実行された更新/挿入/削除 SQL ステートメント』

826 ページの『unread_prefetch_pages - 読み取り不能プリフェッチ・ページ : モニター・エレメント』

832 ページの『user_cpu_time ユーザー CPU 時間』

843 ページの『x_lock_escals 排他ロック・エスカレーション数』

845 ページの『xquery_stmts - 試行された XQuery ステートメント』

event_connheader 論理データ・グループ

event_connmemuse 論理データ・グループ

event_data_value 論理データ・グループ

event_db 論理データ・グループ

626 ページの『pool_async_read_time バッファー・プール非同期読み取り時間』

627 ページの『pool_async_write_time バッファー・プール非同期書き込み時間』

632 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

634 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

635 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

638 ページの『pool_drty_pg_steal_clns - 起動されたバッファー・プール・ビクティム・ページ・クリーナー : モニター・エレメント』

639 ページの『pool_drty_pg_thrsh_clns - 起動されたバッファー・プールしきい値クリーナー : モニター・エレメント』

641 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

643 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

645 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

647 ページの『pool_lsn_gap_clns - 起動されたバッファー・プール・ログ・スペース・クリーナー : モニター・エレメント』

648 ページの『pool_no_victim_buffer - バッファー・プールの非ビクティム・バッファー数 : モニター・エレメント』

649 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

652 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

654 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

655 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

657 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

663 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

671 ページの『post_shrthreshold_hash_joins ポストしきい値ハッシュ結合』

671 ページの『post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント』

675 ページの『prefetch_wait_time - プリフェッチ待ち時間 : モニター・エレメント』

677 ページの『priv_workspace_num_overflows 専用ワークスペースのオーバーフロー回数』

678 ページの『priv_workspace_section_inserts 専用ワークスペース・セクション挿入』

679 ページの『priv_workspace_section_lookups 専用ワークスペース・セクション検索』

679 ページの『priv_workspace_size_top 専用ワークスペースの最大サイズ』

701 ページの『rollback_sql_stmts 試行されたロールバック・ステートメント』

704 ページの『rows_deleted - 削除行数 : モニター・エレメント』

705 ページの『rows_inserted - 挿入行数 : モニター・エレメント』

707 ページの『rows_read - 読み取り行数 : モニター・エレメント』

711 ページの『rows_selected 選択行数』

712 ページの『rows_updated - 更新行数 : モニター・エレメント』

715 ページの『sec_log_used_top 使用された最大 2 次ログ・スペース』

718 ページの『select_sql_stmts 実行された選択 SQL ステートメント』

721 ページの『server_platform サーバーのオペレーティング・システム』

726 ページの『shr_workspace_num_overflows 共有ワークスペースのオーバーフロー回数』

727 ページの『shr_workspace_section_inserts 共有ワークスペース・セクション挿入数』

728 ページの『shr_workspace_section_lookups 共有ワークスペース・セクション検索』

728 ページの『shr_workspace_size_top 最大共有ワークスペース・サイズ』

731 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

744 ページの『static_sql_stmts 試行された静的 SQL ステートメント』

745 ページの『stats_cache_size - 統計キャッシュのサイズ: モニター・エレメント』

746 ページの『stats_fabricate_time - 統計作成アクティビティーに費やされた合計時間: モニター・エレメント』

746 ページの『stats_fabrications - 統計作成の合計数: モニター・エレメント』

763 ページの『sync_runstats - 同期 RUNSTATS アクティビティーの合計数: モニター・エレメント』

764 ページの『sync_runstats_time - 同期 RUNSTATS アクティビティーに費やされた合計時間: モニター・エレメント』

800 ページの『tot_log_used_top 使用された最大合計ログ・スペース』

803 ページの『total_cons データベース活動化以降の接続』

805 ページの『total_hash_joins ハッシュ結合の合計』

806 ページの『total_hash_loops ハッシュ・ループの合計』

807 ページの『total_olap_funcs OLAP 関数の合計数 : モニター・エレメント』

813 ページの『total_sort_time - ソート時間合計 : モニター・エレメント』

814 ページの『total_sorts - ソート合計 : モニター・エレメント』

825 ページの『uid_sql_stmts 実行された更新/挿入/削除 SQL ステートメント』

826 ページの『unread_prefetch_pages - 読み取り不能プリフェッチ・ページ : モニター・エレメント』

843 ページの『x_lock_escals 排他ロック・エスカレーション数』

845 ページの『xquery_stmts - 試行された XQuery ステートメント』

event_dbheader 論理データ・グループ

428 ページの『conn_time データベース接続時刻 : モニター・エレメント』

452 ページの『db_name データベース名』

453 ページの『db_path データベース・パス』

event_dbmemuse 論理データ・グループ

590 ページの『node_number ノード番号』

630 ページの『pool_config_size メモリー・プールの構成済みサイズ』

631 ページの『pool_cur_size メモリー・プールの現行サイズ』

640 ページの『pool_id メモリー・プール ID』

662 ページの『pool_watermark メモリー・プール水準点』

event_deadlock 論理データ・グループ

458 ページの『deadlock_id デッドロック・イベント ID』

459 ページの『deadlock_node デッドロック発生場所のパーティション番号』

476 ページの『dl_conns - デッドロックに関係している接続 : モニター・エレメント』

482 ページの『evmon_activates イベント・モニター活動化回数』

702 ページの『rolled_back_agent_id ロールバックされたエージェント』

703 ページの『rolled_back_appl_id ロールバック・アプリケーション』

703 ページの『rolled_back_participant_no ロールバック参加アプリケーション : モニター・エレメント』

703 ページの『rolled_back_sequence_no ロールバックされたシーケンス番号』

743 ページの『start_time イベント開始時刻』

event_detailed_dlconn 論理データ・グループ

364 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

377 ページの『appl_id - アプリケーション ID :』

379 ページの『appl_id_holding_lk ロックを保持しているアプリケーション ID』

400 ページの『blocking_cursor ブロック・カーソル』

430 ページの『consistency_token パッケージ整合性トークン : モニター・エレメント』

446 ページの『creator アプリケーション作成者』

448 ページの『cursor_name カーソル名』

449 ページの『data_partition_id - データ・パーティション ID : モニター・エレメント』

458 ページの『deadlock_id デッドロック・イベント ID』

459 ページの『deadlock_node デッドロック発生場所のパーティション番号』

482 ページの『evmon_activates イベント・モニター活動化回数』

548 ページの『lock_escalation ロック・エスカレーション : モニター・エレメント』

552 ページの『lock_mode - ロック・モード : モニター・エレメント』

553 ページの『lock_mode_requested 要求されているロック・モード : モニター・エレメント』

554 ページの『lock_node ロック・ノード』

554 ページの『lock_object_name ロック対象名』

555 ページの『lock_object_type - 待機中のロック対象タイプ : モニター・エレメント』

559 ページの『lock_wait_start_time ロック待機開始タイム・スタンプ』

564 ページの『locks_held ロック保持数』

565 ページの『locks_in_list 報告されたロックの回数』

608 ページの『package_name - パッケージ名 : モニター・エレメント』

609 ページの『package_version_id - パッケージ・バージョン : モニター・エレメント』

614 ページの『participant_no デッドロック内の参加者』

615 ページの『participant_no_holding_lk アプリケーションが必要とするオブジェクトのロックを保留する参加者』

716 ページの『section_number - セクション番号 : モニター・エレメント』

719 ページの『sequence_no シーケンス番号 : モニター・エレメント』

720 ページの『sequence_no_holding_lk ロックを保持しているシーケンス番号』

743 ページの『start_time イベント開始時刻』

751 ページの『stmt_operation/operation ステートメント操作 : モニター・エレメント』

757 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

758 ページの『stmt_type ステートメント・タイプ : モニター・エレメント』

766 ページの『table_name - 表名 : モニター・エレメント』

768 ページの『table_schema - 表スキーマ名 : モニター・エレメント』

776 ページの『tablespace_name - 表スペース名 : モニター・エレメント』

event_dlconn 論理データ・グループ

364 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

377 ページの『appl_id - アプリケーション ID :』

379 ページの『appl_id_holding_lk ロックを保持しているアプリケーション ID』

449 ページの『data_partition_id - データ・パーティション ID : モニター・エレメント』

458 ページの『deadlock_id デッドロック・イベント ID』

459 ページの『deadlock_node デッドロック発生場所のパーティション番号』

482 ページの『evmon_activates イベント・モニター活動化回数』

546 ページの『lock_attributes ロック属性 : モニター・エレメント』

547 ページの『lock_count ロック・カウント : モニター・エレメント』

548 ページの『lock_current_mode 変換前の元のロック・モード』

548 ページの『lock_escalation ロック・エスカレーション : モニター・エレメント』

551 ページの『lock_hold_count ロック保留カウント : モニター・エレメント』

552 ページの『lock_mode - ロック・モード : モニター・エレメント』

553 ページの『lock_mode_requested 要求されているロック・モード : モニター・エレメント』

553 ページの『lock_name ロック名 : モニター・エレメント』

554 ページの『lock_node ロック・ノード』

554 ページの『lock_object_name ロック対象名』

555 ページの『lock_object_type - 待機中のロック対象タイプ : モニター・エレメント』

556 ページの『lock_release_flags ロック保留解除フラグ : モニター・エレメント』

559 ページの『lock_wait_start_time ロック待機開始タイム・スタンプ』

614 ページの『participant_no デッドロック内の参加者』

615 ページの『participant_no_holding_lk アプリケーションが必要とするオブジェクトのロックを保留する参加者』

719 ページの『sequence_no シーケンス番号 : モニター・エレメント』

720 ページの『sequence_no_holding_lk ロックを保持しているシーケンス番号』

743 ページの『start_time イベント開始時刻』

766 ページの『table_name - 表名 : モニター・エレメント』

768 ページの『table_schema - 表スキーマ名 : モニター・エレメント』

776 ページの『tablespace_name - 表スペース名 : モニター・エレメント』

818 ページの『tpmon_acc_str TP モニター・クライアント・アカウンティング・ストリング : モニター・エレメント』

819 ページの『tpmon_client_app TP モニター・クライアント・アプリケーション名 : モニター・エレメント』

820 ページの『tpmon_client_userid TP モニター・クライアント・ユーザー ID : モニター・エレメント』

820 ページの『tpmon_client_wkstn TP モニター・クライアント・ワークステーション名 : モニター・エレメント』

event_histogrambin 論理データ・グループ

398 ページの『bin_id ヒストグラム・ビン ID : モニター・エレメント』

401 ページの『bottom ヒストグラム・ビンの最下位 : モニター・エレメント』

522 ページの『histogram_type ヒストグラム・タイプ : モニター・エレメント』

600 ページの『number_in_bin ビン内の数 : モニター・エレメント』

723 ページの『service_class_id サービス・クラス ID : モニター・エレメント』

744 ページの『statistics_timestamp 統計タイム・スタンプ : モニター・エレメント』

800 ページの『top ヒストグラム・ビンの最上位 : モニター・エレメント』

838 ページの『work_action_set_id 作業アクション・セット ID : モニター・エレメント』

839 ページの『work_class_id 作業クラス ID : モニター・エレメント』

840 ページの『workload_id ワークロード ID : モニター・エレメント』

event_log_header 論理データ・グループ

404 ページの『byte_order イベント・データのバイト・オーダー』

416 ページの『codepage_id アプリケーションで使用するコード・ページ ID』

481 ページの『event_monitor_name イベント・モニター名』

597 ページの『num_nodes_in_db2_instance パーティション内のノード数』

721 ページの『server_instance_name サーバー・インスタンス名』

722 ページの『server_prdid - サーバー製品/バージョン ID』

795 ページの『territory_code データベース・テリトリー・コード』

836 ページの『version モニター・データのバージョン』

event_overflow 論理データ・グループ

443 ページの『count イベント・モニター・オーバーフロー数』

504 ページの『first_overflow_time 最初のイベント・オーバーフロー時刻』

543 ページの『last_overflow_time 最後のイベント・オーバーフロー時刻』

590 ページの『node_number ノード番号』

event_qstats 論理データ・グループ

544 ページの『last_wlm_reset 最後にリセットされた時刻 : モニター・エレメント』

686 ページの『queue_assignments_total キュー割り当ての合計 : モニター・エレメント』

686 ページの『queue_size_top キュー・サイズの最上位 : モニター・エレメント』

686 ページの『queue_time_total キュー時間の合計 : モニター・エレメント』

724 ページの『service_subclass_name サービス・サブクラス名 : モニター・エレメント』

725 ページの『service_superclass_name サービス・スーパークラス名 : モニター・エレメント』

744 ページの『statistics_timestamp 統計タイム・スタンプ : モニター・エレメント』

796 ページの『threshold_domain しきい値ドメイン : モニター・エレメント』

797 ページの『threshold_name しきい値名 : モニター・エレメント』

797 ページの『threshold_predicate しきい値述部 : モニター・エレメント』

798 ページの『thresholdid しきい値 ID : モニター・エレメント』

839 ページの『work_action_set_name 作業アクション・セット名 : モニター・エレメント』

839 ページの『work_class_name 作業クラス名 : モニター・エレメント』

event_scstats 論理データ・グループ

355 ページの『act_cpu_time_top – アクティビティーの CPU 時間の最上位 : モニター・エレメント』

357 ページの『act_remapped_in 再マッピングするアクティビティー : モニター・エレメント』

357 ページの『act_remapped_out 再マッピングの際に除外されるアクティビティー : モニター・エレメント』

357 ページの『act_rows_read_top – アクティビティーの読み取り行数の最上位 : モニター・エレメント』

374 ページの『agg_temp_tablespace_top 集約 TEMPORARY 表スペースの最上位 : モニター・エレメント』

420 ページの『concurrent_act_top 並行アクティビティーの最上位 : モニター・エレメント』

421 ページの『concurrent_wlo_top 並行ワークロード・オカレンスの最上位 : モニター・エレメント』

420 ページの『concurrent_connection_top 並行接続の最上位 : モニター・エレメント』

433 ページの『coord_act_aborted_total 打ち切られたコーディネーター・アクティビティーの合計 : モニター・エレメント』

434 ページの『coord_act_completed_total 完了したコーディネーター・アクティビティーの合計 : モニター・エレメント』

434 ページの『coord_act_est_cost_avg コーディネーター・アクティビティーの平均見積コスト : モニター・エレメント』

435 ページの『coord_act_exec_time_avg コーディネーター・アクティビティー平均実行時間 : モニター・エレメント』

436 ページの『coord_act_interarrival_time_avg コーディネーター・アクティビティーの平均到着時間 : モニター・エレメント』

437 ページの『coord_act_lifetime_avg コーディネーター・アクティビティー存続時間の平均 : モニター・エレメント』

438 ページの『coord_act_lifetime_top コーディネーター・アクティビティー存続時間の最上位 : モニター・エレメント』

439 ページの『coord_act_queue_time_avg コーディネーター・アクティビティー・キュー平均時間 : モニター・エレメント』

440 ページの『coord_act_rejected_total リジェクトされたコーディネーター・アクティビティーの合計 : モニター・エレメント』

442 ページの『cost_estimate_top コスト見積もりの最上位 : モニター・エレメント』

details_xml (この XML 文書には、MON_GET_SERVICE_SUBCLASS_DETAILS 表関数の出力の DETAILS 列にレポートされるモニター・エレメントがすべて含まれています。)

544 ページの『last_wlm_reset 最後にリセットされた時刻 : モニター・エレメント』

699 ページの『request_exec_time_avg 要求の平均実行時間 : モニター・エレメント』

710 ページの『rows_returned_top 実際の戻り行数の最上位 : モニター・エレメント』

723 ページの『service_class_id サービス・クラス ID : モニター・エレメント』

724 ページの『service_subclass_name サービス・サブクラス名 : モニター・エレメント』

725 ページの『service_superclass_name サービス・スーパークラス名 : モニター・エレメント』

744 ページの『statistics_timestamp 統計タイム・スタンプ : モニター・エレメント』

794 ページの『temp_tablespace_top TEMPORARY 表スペースの最上位 : モニター・エレメント』

831 ページの『uow_total_time_top - UOW 合計時間の最上位 : モニター・エレメント』

event_start 論理データ・グループ

743 ページの『start_time イベント開始時刻』

event_stmt 論理データ・グループ

364 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

373 ページの『agents_top 作成されたエージェントの数』

377 ページの『appl_id - アプリケーション ID :』

400 ページの『blocking_cursor ブロック・カーソル』

430 ページの『consistency_token パッケージ整合性トークン : モニター・エレメント』

446 ページの『creator アプリケーション作成者』

448 ページの『cursor_name カーソル名』

502 ページの『fetch_count 成功したフェッチの数』

534 ページの『int_rows_deleted 削除された内部行数』

534 ページの『int_rows_inserted 挿入された内部行数』

535 ページの『int_rows_updated 更新された内部行数』

608 ページの『package_name - パッケージ名 : モニター・エレメント』

609 ページの『package_version_id - パッケージ・バージョン : モニター・エレメント』

614 ページの『partial_record 部分レコード : モニター・エレメント』

632 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

634 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

641 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

643 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

652 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

654 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

655 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

657 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

707 ページの『rows_read - 読み取り行数 : モニター・エレメント』

712 ページの『rows_written 書き込み行数』

716 ページの『section_number - セクション番号 : モニター・エレメント』

719 ページの『sequence_no シーケンス番号 : モニター・エレメント』

731 ページの『sort_overflows - ソート・オーバーフロー : モニター・エレメント』

735 ページの『sql_req_id SQL ステートメントの要求 ID』

736 ページの『sqlca SQL 連絡域 (SQLCA)』

743 ページの『start_time イベント開始時刻』

746 ページの『stats_fabricate_time - 統計作成アクティビティーに費やされた合計時間: モニター・エレメント』

751 ページの『stmt_operation/operation ステートメント操作 : モニター・エレメント』

757 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

758 ページの『stmt_type ステートメント・タイプ : モニター・エレメント』

762 ページの『stop_time イベント停止時刻』

764 ページの『sync_runstats_time - 同期 RUNSTATS アクティビティーに費やされた合計時間: モニター・エレメント』

765 ページの『system_cpu_time システム CPU 時間』

813 ページの『total_sort_time - ソート時間合計 : モニター・エレメント』

814 ページの『total_sorts - ソート合計 : モニター・エレメント』

832 ページの『user_cpu_time ユーザー CPU 時間』

event_stmt_history 論理データ・グループ

418 ページの『comp_env_desc コンパイル環境 : モニター・エレメント』

446 ページの『creator アプリケーション作成者』

458 ページの『deadlock_id デッドロック・イベント ID』

459 ページの『deadlock_node デッドロック発生場所のパーティション番号』

482 ページの『evmon_activates イベント・モニター活動化回数』

608 ページの『package_name - パッケージ名 : モニター・エレメント』

609 ページの『package_version_id - パッケージ・バージョン : モニター・エレメント』

614 ページの『participant_no デッドロック内の参加者』

716 ページの『section_number - セクション番号 : モニター・エレメント』

719 ページの『sequence_no シーケンス番号 : モニター・エレメント』

748 ページの『stmt_first_use_time ステートメントの最初の使用時刻』

748 ページの『stmt_history_id ステートメント履歴 ID』

749 ページの『stmt_invocation_id ステートメント呼び出し ID : モニター・エレメント』

749 ページの『stmt_isolation ステートメント分離』

750 ページの『stmt_last_use_time ステートメント最終使用時刻 : モニター・エレメント』

750 ページの『stmt_lock_timeout ステートメント・ロック・タイムアウト : モニター・エレメント』

751 ページの『stmt_nest_level ステートメント・ネスト・レベル : モニター・エレメント』

753 ページの『stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID』

754 ページの『stmt_query_id ステートメント照会 ID : モニター・エレメント』

755 ページの『stmt_source_id ステートメント・ソース ID』

757 ページの『stmt_text - SQL ステートメント・テキスト : モニター・エレメント』

758 ページの『stmt_type ステートメント・タイプ : モニター・エレメント』

event_subsection 論理データ・グループ

364 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

591 ページの『num_agents ステートメントで作動しているエージェントの数』

614 ページの『partial_record 部分レコード : モニター・エレメント』

741 ページの『ss_exec_time サブセクション実行経過時間』

741 ページの『ss_node_number サブセクション・ノード番号』

742 ページの『ss_number サブセクション番号』

742 ページの『ss_sys_cpu_time サブセクションに使用されたシステム CPU 時間』

743 ページの『ss_usr_cpu_time サブセクションに使用されたユーザー CPU 時間』

822 ページの『tq_max_send_spills 表キュー・バッファー・オーバーフローの最大数』

823 ページの『tq_rows_read 表キューから読み取られた行数』

823 ページの『tq_rows_written 表キューに書き込まれた行数』

824 ページの『tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数 : モニター・エレメント』

event_table 論理データ・グループ

449 ページの『data_object_pages データ・オブジェクト・ページ数』

449 ページの『data_partition_id - データ・パーティション ID : モニター・エレメント』

481 ページの『event_time イベント時刻』

482 ページの『evmon_activates イベント・モニター活動化回数』

483 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

527 ページの『index_object_pages 索引オブジェクト・ページ数』

544 ページの『lob_object_pages LOB オブジェクト・ページ数』

572 ページの『long_object_pages 長いオブジェクト・ページ数』

607 ページの『overflow_accesses - オーバーフロー・レコードへのアクセス : モニター・エレメント』

610 ページの『page_reorgs ページ再編成』

614 ページの『partial_record 部分レコード : モニター・エレメント』

707 ページの『rows_read - 読み取り行数 : モニター・エレメント』

712 ページの『rows_written 書き込み行数』

766 ページの『table_name - 表名 : モニター・エレメント』

768 ページの『table_schema - 表スキーマ名 : モニター・エレメント』

769 ページの『table_type - 表タイプ : モニター・エレメント』

event_tablespace 論理データ・グループ

465 ページの『direct_read_reqs - 直接読み取り要求 : モニター・エレメント』

467 ページの『direct_read_time - 直接読み取り時間 : モニター・エレメント』

468 ページの『direct_reads - データベースからの直接読み取り : モニター・エレメント』

470 ページの『direct_write_reqs - 直接書き込み要求 : モニター・エレメント』

471 ページの『direct_write_time - 直接書き込み時間 : モニター・エレメント』

473 ページの『direct_writes - データベースへの直接書き込み : モニター・エレメント』

481 ページの『event_time イベント時刻』

482 ページの『evmon_activates イベント・モニター活動化回数』

483 ページの『evmon_flushes イベント・モニター・フラッシュ回数』

502 ページの『files_closed - クローズしたデータベース・ファイル : モニター・エレメント』

614 ページの『partial_record 部分レコード : モニター・エレメント』

621 ページの『pool_async_data_read_reqs - バッファー・プール非同期読み取り要求 : モニター・エレメント』

622 ページの『pool_async_data_reads バッファー・プール非同期データ読み取り : モニター・エレメント』

623 ページの『pool_async_data_writes - バッファー・プール非同期データ書き込み : モニター・エレメント』

624 ページの『pool_async_index_read_reqs - バッファー・プール非同期索引読み取り要求 : モニター・エレメント』

624 ページの『pool_async_index_reads - バッファー・プール非同期索引読み取り : モニター・エレメント』

625 ページの『pool_async_index_writes - バッファー・プール非同期索引書き込み : モニター・エレメント』

626 ページの『pool_async_read_time バッファー・プール非同期読み取り時間』

627 ページの『pool_async_write_time バッファー・プール非同期書き込み時間』

632 ページの『pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント』

634 ページの『pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント』

635 ページの『pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント』

641 ページの『pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント』

643 ページの『pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント』

645 ページの『pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント』

648 ページの『pool_no_victim_buffer - バッファー・プールの非ビクティム・バッファー数 : モニター・エレメント』

649 ページの『pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント』

652 ページの『pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント』

654 ページの『pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント』

655 ページの『pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント』

657 ページの『pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント』

663 ページの『pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント』

776 ページの『tablespace_name - 表スペース名 : モニター・エレメント』

event_thresholdviolations 論理データ・グループ

358 ページの『activate_timestamp タイム・スタンプの活動化 : モニター・エレメント』

359 ページの『activity_collected 収集されたアクティビティー : モニター・エレメント』

360 ページの『activity_id アクティビティー ID : モニター・エレメント』

364 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

377 ページの『appl_id - アプリケーション ID :』

441 ページの『coord_partition_num コーディネーター・パーティション番号 : モニター・エレメント』

463 ページの『destination_service_class_id 宛先サービス・クラス ID : モニター・エレメント』

733 ページの『source_service_class_id ソース・サービス・クラス ID : モニター・エレメント』

795 ページの『threshold_action しきい値アクション : モニター・エレメント』

796 ページの『threshold_maxvalue しきい値最大値 : モニター・エレメント』

797 ページの『threshold_predicate しきい値述部 : モニター・エレメント』

797 ページの『threshold_queuesize しきい値キュー・サイズ : モニター・エレメント』

798 ページの『thresholdid しきい値 ID : モニター・エレメント』

799 ページの『time_of_violation 違反時刻 : モニター・エレメント』

828 ページの『uow_id 作業単位 ID : モニター・エレメント』

event_wlstats 論理データ・グループ

355 ページの『act_cpu_time_top – アクティビティーの CPU 時間の最上位 : モニター・エレメント』

357 ページの『act_rows_read_top – アクティビティーの読み取り行数の最上位 : モニター・エレメント』

420 ページの『concurrent_wlo_act_top 並行 WLO アクティビティーの最上位 : モニター・エレメント』

421 ページの『concurrent_wlo_top 並行ワークロード・オカレンスの最上位 : モニター・エレメント』

433 ページの『coord_act_aborted_total 打ち切られたコーディネーター・アクティビティーの合計 : モニター・エレメント』

434 ページの『coord_act_completed_total 完了したコーディネーター・アクティビティーの合計 : モニター・エレメント』

434 ページの『coord_act_est_cost_avg コーディネーター・アクティビティーの平均見積コスト : モニター・エレメント』

435 ページの『coord_act_exec_time_avg コーディネーター・アクティビティー平均実行時間 : モニター・エレメント』

436 ページの『coord_act_interarrival_time_avg コーディネーター・アクティビティーの平均到着時間 : モニター・エレメント』

437 ページの『coord_act_lifetime_avg コーディネーター・アクティビティー存続時間の平均 : モニター・エレメント』

438 ページの『coord_act_lifetime_top コーディネーター・アクティビティー存続時間の最上位 : モニター・エレメント』

439 ページの『coord_act_queue_time_avg コーディネーター・アクティビティー・キュー平均時間 : モニター・エレメント』

440 ページの『coord_act_rejected_total リジェクトされたコーディネーター・アクティビティーの合計 : モニター・エレメント』

442 ページの『cost_estimate_top コスト見積もりの最上位 : モニター・エレメント』

details_xml (この XML 文書には、MON_GET_WORKLOAD_DETAILS 表関数の出力の DETAILS 列にレポートされるモニター・エレメントがすべて含まれています。)

544 ページの『last_wlm_reset 最後にリセットされた時刻 : モニター・エレメント』

562 ページの『lock_wait_time_top – ロック待機時間の最上位 : モニター・エレメント』

710 ページの『rows_returned_top 実際の戻り行数の最上位 : モニター・エレメント』

744 ページの『statistics_timestamp 統計タイム・スタンプ : モニター・エレメント』

831 ページの『uow_total_time_top - UOW 合計時間の最上位 : モニター・エレメント』

794 ページの『temp_tablespace_top TEMPORARY 表スペースの最上位 : モニター・エレメント』

838 ページの『wlo_completed_total 完了したワークロード・オカレンスの合計 : モニター・エレメント』

840 ページの『workload_id ワークロード ID : モニター・エレメント』

841 ページの『workload_name ワークロード名 : モニター・エレメント』

event_wcstats 論理データ・グループ

355 ページの『act_cpu_time_top – アクティビティーの CPU 時間の最上位 : モニター・エレメント』

357 ページの『act_rows_read_top – アクティビティーの読み取り行数の最上位 : モニター・エレメント』

358 ページの『act_total アクティビティーの合計 : モニター・エレメント』

434 ページの『coord_act_est_cost_avg コーディネーター・アクティビティーの平均見積コスト : モニター・エレメント』

435 ページの『coord_act_exec_time_avg コーディネーター・アクティビティー平均実行時間 : モニター・エレメント』

436 ページの『coord_act_interarrival_time_avg コーディネーター・アクティビティーの平均到着時間 : モニター・エレメント』

437 ページの『coord_act_lifetime_avg コーディネーター・アクティビティー存続時間の平均 : モニター・エレメント』

438 ページの『coord_act_lifetime_top コーディネーター・アクティビティー存続時間の最上位 : モニター・エレメント』

439 ページの『coord_act_queue_time_avg コーディネーター・アクティビティー・キュー平均時間 : モニター・エレメント』

442 ページの『cost_estimate_top コスト見積もりの最上位 : モニター・エレメント』

544 ページの『last_wlm_reset 最後にリセットされた時刻 : モニター・エレメント』

710 ページの『rows_returned_top 実際の戻り行数の最上位 : モニター・エレメント』

744 ページの『statistics_timestamp 統計タイム・スタンプ : モニター・エレメント』

794 ページの『temp_tablespace_top TEMPORARY 表スペースの最上位 : モニター・エレメント』

838 ページの『work_action_set_id 作業アクション・セット ID : モニター・エレメント』

839 ページの『work_action_set_name 作業アクション・セット名 : モニター・エレメント』

839 ページの『work_class_id 作業クラス ID : モニター・エレメント』

839 ページの『work_class_name 作業クラス名 : モニター・エレメント』

event_xact 論理データ・グループ

364 ページの『agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント』

377 ページの『appl_id - アプリケーション ID :』

549 ページの『lock_escals - ロック・エスカレーション数 : モニター・エレメント』

560 ページの『lock_wait_time - ロック待機中の時間 : モニター・エレメント』

564 ページの『locks_held_top ロック保持最大数』

614 ページの『partial_record 部分レコード : モニター・エレメント』

677 ページの『prev_uow_stop_time 直前の作業単位完了タイム・スタンプ』

707 ページの『rows_read - 読み取り行数 : モニター・エレメント』

712 ページの『rows_written 書き込み行数』

719 ページの『sequence_no シーケンス番号 : モニター・エレメント』

765 ページの『system_cpu_time システム CPU 時間』

818 ページの『tpmon_acc_str TP モニター・クライアント・アカウンティング・ストリング : モニター・エレメント』

819 ページの『tpmon_client_app TP モニター・クライアント・アプリケーション名 : モニター・エレメント』

820 ページの『tpmon_client_userid TP モニター・クライアント・ユーザー ID : モニター・エレメント』

820 ページの『tpmon_client_wkstn TP モニター・クライアント・ワークステーション名 : モニター・エレメント』

829 ページの『uow_log_space_used 使用されている作業単位ログ・スペース』

829 ページの『uow_start_time 作業単位開始タイム・スタンプ』

830 ページの『uow_status 作業単位の状況』

830 ページの『uow_stop_time 作業単位停止タイム・スタンプ : モニター・エレメント』

832 ページの『user_cpu_time ユーザー CPU 時間』

843 ページの『x_lock_escals 排他ロック・エスカレーション数』

lock 論理データ・グループ

449 ページの『data_partition_id - データ・パーティション ID : モニター・エレメント』

546 ページの『lock_attributes ロック属性 : モニター・エレメント』

547 ページの『lock_count ロック・カウント : モニター・エレメント』

548 ページの『lock_current_mode 変換前の元のロック・モード』

548 ページの『lock_escalation ロック・エスカレーション : モニター・エレメント』

551 ページの『lock_hold_count ロック保留カウント : モニター・エレメント』

552 ページの『lock_mode - ロック・モード : モニター・エレメント』

553 ページの『lock_name ロック名 : モニター・エレメント』

554 ページの『lock_object_name ロック対象名』

555 ページの『lock_object_type - 待機中のロック対象タイプ : モニター・エレメント』

556 ページの『lock_release_flags ロック保留解除フラグ : モニター・エレメント』

557 ページの『lock_status - ロック状況 : モニター・エレメント』

590 ページの『node_number ノード番号』

766 ページの『table_file_id - 表ファイル ID : モニター・エレメント』

766 ページの『table_name - 表名 : モニター・エレメント』

768 ページの『table_schema - 表スキーマ名 : モニター・エレメント』

776 ページの『tablespace_name - 表スペース名 : モニター・エレメント』

sqlca 論理データ・グループ

sqlcabc

sqlcaid

sqlcode

sqlerrd

sqlerrmc

sqlerrml

sqlerrp

sqlstate

sqlwarn

COLLECT ACTIVITY DATA 設定の影響を受ける論理データ・グループ

以下の表は、サービス・サブクラス、ワークロード、作業クラス (作業アクションを介するもの)、およびしきい値を含むすべてのタイプの WLM オブジェクトにおいて、COLLECT ACTIVITY DATA のさまざまなオプションが指定された場合にどの論理データ・グループが収集されるかを示しています。

表 76. COLLECT ACTIVITY DATA 設定

COLLECT ACTIVITY DATA の設定	**収集される論理データ・グループ**
NONE	none
WITHOUT DETAILS	event_activity
WITH DETAILS	event_activity event_activitystmt
WITH DETAILS AND VALUES	event_activity event_activitystmt event_activityvals

第 15 章 データベース・システム・モニターのエレメント

モニター・エレメントによって収集されるデータの説明。

システム・モニターが戻すモニター・エレメントは次のように分類できます。

- モニターされるデータベース・マネージャー、アプリケーション、またはデータベース接続の**識別**。
- システムの**構成**に最初に必要とされるデータ。
- データベース、アプリケーション、表、またはステートメントを含むさまざまなレベルのデータベース・**アクティビティー**。この情報を使用して、アクティビティーのモニター、問題判別、およびパフォーマンス分析を行うことができます。この情報を構成にも使用することができます。
- **DB2 Connect** アプリケーションに関する情報。この中には、ゲートウェイで実行中の DCS アプリケーション、実行中の SQL ステートメント、およびデータベース接続に関する情報が含まれます。
- **フェデレーテッド・データベース・システム**に関する情報。これには、 DB2 フェデレーテッド・システム内で実行中の各アプリケーションからデータ・ソースへのすべてのアクセスに関する情報、およびフェデレーテッド・サーバー・インスタンス内で実行中の特定のアプリケーションからデータ・ソースへのアクセスに関する情報が含まれます。

モニター・エレメントは次の標準形式で記述されます。

エレメント ID
エレメントの名前。データ・ストリームをそのまま構文解析すると、エレメント ID が大文字となり、 SQLM_ELM_ の接頭部が付きます。

エレメント・タイプ
モニター・エレメントが戻す情報のタイプ。例えば db2start_time モニター・エレメントはタイム・スタンプを戻します。

スナップショット・モニター情報
モニター・エレメントがスナップショット・モニター情報を戻す場合は、次のフィールドがある表が記載されています。

- *スナップショット・レベル* : スナップショット・モニターでキャプチャー可能な情報レベル。例えば appl_status モニター・エレメントは、アプリケーション・レベルおよびロック・レベルでの情報を戻します。
- *論理データ・グループ* : キャプチャーされたスナップショット情報が戻される論理データ・グループ。データ・ストリームをそのまま構文解析すると、論理データ・グループ ID が大文字となり、 SQLM_ELM_ の接頭部が付きます。例えば、appl_status モニター・エレメントは、appl_id_info グループおよび appl_lock_list グループの情報を戻します。
- モニター・スイッチ : この情報を取得するために設定が必要なシステム・モニター・スイッチ。スイッチが「基本」の場合は、モニター・エレメントのデータが常に収集されます。

イベント・モニター情報

モニター・エレメントがイベント・モニターによって収集される場合は、次のフィールドがある表が記載されています。

- *イベント・タイプ*：イベント・モニターで収集可能な情報のレベル。この情報を収集するには、このイベント・タイプを使用してイベント・モニターを作成する必要があります。例えば appl_status モニター・エレメントは、「接続」イベント・モニターで収集されます。
- *論理データ・グループ*：取り込まれたイベント情報が戻される論理データ・グループ。データ・ストリームをそのまま構文解析すると、論理データ・グループ ID が大文字となり、 SQLM_ELM_ の接頭部が付きます。例えば appl_status モニター・エレメントは、 event_conn グループの情報を戻します。
- モニター・スイッチ：この情報を取得するために設定が必要なシステム・モニター・スイッチ。イベント・モニターの場合、イベント・データの収集を制約できるモニター・スイッチは TIMESTAMP スイッチのみです。このフィールドにダッシュが示されている場合は、モニター・エレメントのデータが常に収集されます。

使用法 データベース・システムをモニターする際の、モニター・エレメントによって収集された情報の使用方法に関する情報。

acc_curs_blk 受け入れられたブロック・カーソル要求

入出力ブロック要求が受け入れられた回数。

エレメント ID

acc_curs_blk

エレメント・タイプ

カウンター

表 77. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

表 78. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	-

使用法 このエレメントと *rej_curs_blk* を組み合わせて使用すると、受け入れられたブロッキング要求、リジェクトされたブロッキング要求、またはその両方のパーセンテージを計算できます。

この情報を使用して構成パラメーターを調整する方法については、『*rej_curs_blk*』を参照してください。

act_aborted_total - 異常終了したアクティビティーの合計 : モニター・エレメント

エラーで終了した、任意のネスト・レベルのコーディネーター・アクティビティーの合計数。サービス・クラスでは、アクティビティーが異常終了前に REMAP ACTIVITY アクションで別のサービス・サブクラスに再マップされた場合、そのアクティビティーのカウントは、異常終了時のサブクラスでの合計にのみ含められます。

表 79. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 80. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このエレメントを使用して、システム上のアクティビティーが正常に完了しているかを知ることができます。アクティビティーは、取り消し、エラー、または反応的しきい値のために異常終了することがあります。

act_completed_total - 完了したアクティビティーの合計 : モニター・エレメント

正常に完了した、任意のネスト・レベルのコーディネーター・アクティビティーの合計数。サービス・クラスでは、アクティビティーが完了前に REMAP ACTIVITY アクションで別のサブクラスに再マップされた場合、そのアクティビティーのカウントは、完了時のサブクラスでの合計にのみ含められます。

表 81. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 82. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このエレメントを使用して、システム内のアクティビティーのスループットを判別します。

act_cpu_time_top – アクティビティーの CPU 時間の最上位 : モニター・エレメント

サービス・クラス、ワークロード、または作業クラスでの、すべてのネスト・レベルにおけるアクティビティーで使用されるプロセッサー時間の最高水準点。

アクティビティーが実行されるサービス・クラスまたはワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。要求メトリックが使用可能になっている場合にのみ、アクティビティーはこの最高水準点に寄与します。

サービス・クラスでは、REMAP ACTIVITY アクションを使用してサービス・サブクラス間でアクティビティーを再マップすると、新規の最高水準点に達した場合、アクティビティーが完了するサービス・サブクラスの act_cpu_time_top 最高水準点のみが更新されます。アクティビティーがマップされるものの完了していない他のサービス・サブクラスの act_cpu_time_top 最高水準点は、影響を受けません。

表 83. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

このエレメントを使用して、収集された時間間隔にサービス・クラス、ワークロード、または作業クラス用のパーティションでアクティビティーによって使用された最大プロセッサー時間を判別することができます。

act_exec_time アクティビティー実行時間 : モニター・エレメント

このパーティションで実行するために費やされた時間 (マイクロ秒単位)。カーソルの場合、実行時間はオープン、フェッチ、およびクローズの時間を組み合わせたものです。カーソルのアイドル時間は実行時間にカウントされません。ルーチンの場合、実行時間はルーチン呼び出しの開始から終了までです。ルーチンの完了後にそのルーチンによって (結果セットを戻すために) オープンされたままになっているカーソルの存続期間は、ルーチンの実行時間にカウントされません。他のすべてのアクティビティーの場合、実行時間は開始時刻から停止時刻までの時間です。どの場合でも、実行時間には、初期化されている時間またはキューに入れられている時間は含まれません。

表 84. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

使用法

このエレメントを単独で使用すると、パーティションごとに DB2 によるアクティビティーの実行に費やされた経過時間を知ることができます。このエレメントは、**time_started** および **time_completed** モニター・エレメントと一緒にコーディネーター・パーティションで使用して、カーソル・アクティビティーにおけるアイドル時間を計算することもできます。以下の公式を使用できます。

```
カーソルのアイドル時間 = (time_completed - time_started) - act_exec_time
```

act_rejected_total - リジェクトされたアクティビティーの合計 : モニター・エレメント

実行が許可されず、リジェクトされた任意のネスト・レベルのコーディネーター・アクティビティーの合計数。

表 85. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 86. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このエレメントを使用すると、実行を阻止する予測しきい値および作業アクションが有効であるかどうか、および、それらの制限が大きすぎないかどうかを判別する助けになります。

act_remapped_in 再マッピングするアクティビティー : モニター・エレメント

最後のリセット以降に、このサービス・サブクラスに再マッピングするアクティビティーの数。

表 87. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-

使用法

このカウントを使用して、サービス・サブクラスへのアクティビティーの再マップが期待どおりに行われているかを判別します。

act_remapped_out 再マッピングの際に除外されるアクティビティー : モニター・エレメント

最後のリセット以降、再マッピングの際にこのサービス・サブクラスから除外されるアクティビティーの数。

表 88. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-

使用法

このカウントを使用して、サービス・サブクラスからのアクティビティーの再マップが期待どおりに行われているかを判別します。

act_rows_read_top – アクティビティーの読み取り行数の最上位 : モニター・エレメント

サービス・クラス、ワークロード、または作業クラスでの、すべてのネスト・レベルにおけるアクティビティーによって読み取られる行数の最高水準点。

アクティビティーが実行されるサービス・クラスまたはワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。要求メトリックが使用可能になっている場合にのみ、アクティビティーはこの最高水準点に寄与します。

サービス・クラスでは、REMAP ACTIVITY アクションを使用してサービス・サブクラス間でアクティビティーを再マップすると、新規の最高水準点に達した場合、アクティビティーが完了するサービス・サブクラスの act_rows_read_top 最高水準点のみが更新されます。アクティビティーがマップされるものの完了していないサービス・サブクラスの act_rows_read_top 最高水準点は、影響を受けません。

表 89. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

このエレメントを使用して、収集された時間間隔にサービス・クラス、ワークロード、または作業クラス用のパーティションでアクティビティーによって読み取られる最大行数を判別することができます。

act_total アクティビティーの合計 : モニター・エレメント

最後にリセットしてから指定した作業クラスに対応する作業アクションが適用された、任意のネスト・レベルのアクティビティーの合計数。

表 90. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wcstats	-

使用法

作業クラスに関連付けられた 1 つ以上の作業アクションがアクティビティーに適用されるたびに、この作業クラスのカウンターが更新されます。**act_total** モニター・エレメントを使用すると、このカウンターが公開されます。このカウンターを使用して、作業アクション・セットの有効性 (例えば、アクションが適用されているアクティビティーの数) を判断できます。また、システム上のアクティビティーのさまざまなタイプを理解するためにも使用できます。

activate_timestamp タイム・スタンプの活動化 : モニター・エレメント

イベント・モニターがアクティブにされた時刻。

表 91. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-
アクティビティー	event_activitystmt	-
アクティビティー	event_activityvals	-
しきい値違反	event_thresholdviolations	-

使用法

このエレメントを使用して、上記のイベント・タイプで戻された情報を関連付けます。

active_hash_joins - アクティブ・ハッシュ結合

現在実行中でメモリーを消費しているハッシュ結合の合計数

表 92. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	-

active_olap_funcs アクティブ OLAP 関数 : モニター・エレメント

現在実行中でソート・ヒープ・メモリーを消費している OLAP 関数の合計数

表 93. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	-

スナップショット・モニターの場合、このカウンターはリセットできます。

active_sorts アクティブ・ソート

現在ソート・ヒープが割り振られているデータベース内のソート数。

表 94. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 この値と *sort_heap_allocated* を組み合わせて使用すると、各ソートで使用される平均ソート・ヒープ・スペースを判別できます。使用されている平均ソート・ヒープと比較して、 *sortheap* 構成パラメーターが非常に大きい場合は、このパフォーマンス値を低くできます。

この値には、関連操作で作成された一時表のソートのヒープが含まれます。

activity_collected 収集されたアクティビティー : モニター・エレメント

このエレメントは、しきい値の違反が発生した場合にアクティビティー・イベント・モニター・レコードが収集されるかどうかを示します。

表 95. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-

使用法

このエレメントを使用すると、しきい値を違反したアクティビティーのアクティビティー・イベントがアクティビティー・イベント・モニターに書き込まれるかどうかを判別できます。

アクティビティーが完了またはアボートし、その時点でアクティビティー・イベント・モニターがアクティブである場合、このモニター・エレメントの値が「Y」である場合には、このしきい値に違反したアクティビティーは収集されます。このモニター・エレメントの値が「N」である場合、それは収集されません。

activity_id アクティビティー ID : モニター・エレメント

特定の作業単位内のアプリケーションのアクティビティーを一意的に識別するカウンター。

表 96. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

表 97. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
アクティビティー	event_activity	-
アクティビティー	event_activitystmt	-
アクティビティー	event_activityvals	-
しきい値違反	event_thresholdviolations	-

使用法

このエレメントを他のアクティビティー履歴エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。

アクティビティーをその作業単位外から一意的に識別するには、**activity_id** および **uow_id** と、**appl_id** または **agent_id** のいずれかを組み合わせて使用してください。

activity_secondary_id アクティビティー 2 次 ID : モニター・エレメント

このエレメントの値は、同じアクティビティーに関してアクティビティー・レコードが書き込まれるたびに増分されます。例えば、アクティビティー・レコードが、WLM_CAPTURE_ACTIVITY_IN_PROGRESS プロシージャーを呼び出した結果として 1 回目に書き込まれ、アクティビティーが終了した時に 2 回目に書き込まれた場合、エレメントの値は、最初のレコードについては 0、2 番目のレコードについては 1 となります。

表 98. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-
アクティビティー	event_activitystmt	-
アクティビティー	event_activityvals	-

使用法

このエレメントを **activity_id**、**uow_id**、および **appl_id** モニター・エレメントと一緒に使用すると、同一のアクティビティーに関する情報がアクティビティー・イベント・モニターに複数回書き込まれた場合にアクティビティー・レコードを一意的に識別できます。

例えば、以下の場合には、アクティビティーに関する情報がアクティビティー・イベント・モニターに 2 回送信されます。

- WLM_CAPTURE_ACTIVITY_IN_PROGRESS ストアード・プロシージャーを使用して、実行中のアクティビティーに関する情報をキャプチャーした場合
- アクティビティーが関連付けられているサービス・クラス上で COLLECT ACTIVITY DATA 文節を指定したために、そのアクティビティーの完了時にそのアクティビティーに関する情報を収集した場合。

activity_state - アクティビティーの状態 : モニター・エレメント

アクティビティーの現在の状態。

表 99. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このモニター・エレメントを使用して、アクティビティーの現在の状況を判別します (例えば、アクティビティーがキューに入れられている、あるいはクライアントからの入力を待機している、など)。可能な値は以下のとおりです。

- CANCEL_PENDING
- EXECUTING
- IDLE
- INITIALIZING
- QP_CANCEL_PENDING
- QP_QUEUED
- QUEUED
- TERMINATING

- UNKNOWN

activity_type アクティビティー・タイプ : モニター・エレメント

アクティビティーのタイプ。

表 100. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

表 101. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

使用法

可能な値は以下のとおりです。

- LOAD
- READ_DML
- WRITE_DML
- DDL
- CALL
- OTHER

activitytotaltime_threshold_id - アクティビティー合計時間しきい値 ID : モニター・エレメント

アクティビティーに適用されていた ACTIVITYTOTALTIME しきい値の ID。

表 102. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、 ACTIVITYTOTALTIME しきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

activitytotaltime_threshold_value - アクティビティー合計時間しきい値 : モニター・エレメント

アクティビティー・エントリー時からの ACTIVITYTOTALTIME の合計。アクティビティーがこのタイム・スタンプに達してもまだ実行している場合、しきい値に違反したことになります。

表 103. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、ACTIVITYTOTALTIME しきい値がアクティビティーに適用されている場合、その値を判別します。

activitytotaltime_threshold_violated - アクティビティー合計時間しきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが ACTIVITYTOTALTIME しきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 104. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、アクティビティーに適用されていた ACTIVITYTOTALTIME しきい値にアクティビティーが違反したかどうかを判別します。

address - 接続の開始元となった IP アドレス

アクティビティー接続の開始元となった IP アドレス。

表 105. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

使用法

これを使用して、アクティビティー接続の開始元となった IP アドレスを識別できます。セキュア・ドメイン名は、IP アドレスに変換されて表示されます。

agent_id アプリケーション・ハンドル (エージェント ID) : モニター・エレメント

システム全体での、アプリケーションのユニーク ID。単一パーティションのデータベースの場合は、この ID は 16 ビット・カウンターで構成されます。複数パーティションのデータベースでは、この ID はコーディネーター・パーティション番号と 16 ビット・カウンターが連結されたもので構成されます。さらに、アプリケーションが 2 次接続を行う可能性のあるパーティションには、すべて同一の ID が使用されます。

表 106. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	常に収集される
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	常に収集される
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

表 107. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	基本
ロック	appl_lock_list	基本
DCS アプリケーション	dcs_appl_info	基本

表 108. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
作業単位	-	-
接続	event_connheader	-
ステートメント	event_stmt	-
ステートメント	event_subsection	-
デッドロック[1]	event_dlconn	-
詳細付きデッドロック[1]	event_detailed_dlconn	-

表 108. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-
アクティビティー	event_activity	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

アプリケーション・ハンドルは、エージェント ID とも呼ばれ、これを使用するとアクティブ・アプリケーションを一意的に識別できます。

注: **agent_id** モニター・エレメントは、使用する DB2 のバージョンによって動作が異なります。バージョン SQLM_DBMON_VERSION1 または SQLM_DBMON_VERSION2 の DB2 から DB2 (バージョン 5 またはそれ以上) のデータベースへスナップショットをとる時、戻される **agent_id** はアプリケーション ID として使用できず、アプリケーションを提供するエージェントの **agent_pid** として有効です。このような場合、**agent_id** は以前のリリースとの互換性のために引き続き戻されますが、内部的には、 DB2 データベース・サーバーはその値を **agent_id** として認識しません。

この値は、エージェント ID を必要とする GET SNAPSHOT コマンドへの入力として、またはアプリケーション・ハンドルを必要とするモニター表関数への入力として使用できます。

またイベント・トレースを読み取るとき、イベント・レコードを指定のアプリケーションと一致させるために使用できます。

FORCE APPLICATION コマンドまたは API への入力として使用することができます。マルチノード・システムでは、アプリケーションが接続されているノードから、このコマンドを出すことができます。効力範囲はグローバルです。

agent_id_holding_lock ロックを保持しているエージェント ID

このアプリケーションが待機しているロックを保持しているエージェントのアプリケーション・ハンドル。この情報を取得するには、ロック・モニター・グループをオンにする必要があります。

表 109. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ロック
ロック	appl_lock_list	ロック
ロック	lock_wait	ロック

使用法 このエレメントは、リソースの競合状態にあるアプリケーションの判別に利用できます。

このエレメントが 0 (ゼロ) で、アプリケーションがロックを待機中の場合は、ロックが未確定トランザクションによって保持されていることを示します。 appl_id_holding_lk または コマンド行プロセッサー LIST INDOUBT TRANSACTIONS コマンドのいずれかを使用して (未確定となったときにトランザクションを処理していた CICS エージェントのアプリケーション ID を表示します)、未確定トランザクションを識別してから、コミットまたはロールバックを行います。

このアプリケーションが待機している 1 つのオブジェクトに対して、複数のアプリケーションが共有ロックを保有していることがあるので注意してください。アプリケーションが保留しているロックのタイプについては、『lock_mode』を参照してください。アプリケーションのスナップショットをとる場合は、オブジェクトのロックを保留しているエージェント ID のうち戻されるのは 1 つだけです。ロックのスナップショットを取れば、オブジェクトのロックを保留しているすべてのエージェント ID を識別できます。

agent_pid エンジン・ディスパッチ可能単位 (EDU) ID : モニター・エレメント

エージェントのエンジン・ディスパッチ可能単位 (EDU) のユニーク ID。Linux オペレーティング・システムを除いて、EDU ID はスレッド ID にマップされます。Linux オペレーティング・システムでは、EDU ID は DB2 生成のユニーク ID です。

表 110. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	agent	ステートメント

表 111. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-

使用法

このエレメントを使用すると、データベース・システム・モニター情報をシステム・トレースなどの他の診断情報のソースにリンクできます。また、このエレメントを使用すると、データベース・アプリケーションの処理を行うエージェントがシステム・リソースをどのように使用しているのかをモニターできます。

agent_status DCS アプリケーション・エージェント

接続コンセントレーター環境では、この値は、関連エージェントを現在持っているアプリケーションを示します。

エレメント ID

agent_status

エレメント・タイプ

情報

表 112. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl_info	基本

使用法 次の値があります。

- SQLM_AGENT_ASSOCIATED

  このアプリケーションに代わって作業中のエージェントがアプリケーションに関連付けられています。

- SQLM_AGENT_NOT_ASSOCIATED

  このアプリケーションに代わって作業していたエージェントは、もはやアプリケーションには関連付けられていません。現在はほかのアプリケーションで使用されています。この後、関連エージェントがないままこのアプリケーションの作業が行われると、エージェントが再び関連付けられます。

agent_sys_cpu_time エージェントが使用したシステム CPU 時間

データベース・マネージャーのエージェント・プロセスで使用されたシステム CPU 時間の合計 (秒単位およびマイクロ秒単位)。

エレメント ID

agent_sys_cpu_time

エレメント・タイプ

時間

表 113. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	タイム・スタンプ

アプリケーション・レベルでのスナップショット・モニターの場合、このカウンターはリセットできます。その他のレベルではこのカウンターはリセットできません。

使用法 このエレメントと CPU 時間に関連する他のエレメントを組み合わせて使用すると、アプリケーション内のアクティビティーのレベルがわかります。また、さらに調整するとその効果が得られる可能性があるアプリケーションを識別できます。

このエレメントには、SQL および非 SQL のステートメントの両者の CPU 時間、およびその他の unfenced ユーザー定義関数 (UDF) の CPU 時間が含まれます。

システム CPU は、システム呼び出しに要した時間を示します。ユーザー CPU は、データベース・マネージャーのコードを実行するのに要した時間を示します。

注: ユーザーのオペレーティング・システムでこの情報が利用できないとき、このエレメントは 0 に設定されます。

agent_usr_cpu_time エージェントが使用したユーザー CPU 時間

データベース・マネージャーのエージェント・プロセスで使用された CPU 時間の合計 (秒およびマイクロ秒単位)。

エレメント ID

agent_usr_cpu_time

エレメント・タイプ

時間

表 114. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントとその他の CPU 時間関連のエレメントを組み合わせて使用すると、 CPU を大量に使用しているアプリケーションや照会を識別できます。

このカウンターには、SQL および非 SQL のステートメントに要した時間のほか、アプリケーションが実行した unfenced ユーザー定義関数 (UDF) およびストアード・プロシージャーも含まれます。

システム CPU は、システム呼び出しに要した時間を示します。ユーザー CPU は、データベース・マネージャーのコードを実行するのに要した時間を示します。

注: 使用しているオペレーティング・システムでこの情報を得られない場合は、このエレメントはゼロ (0) を戻します。

agent_wait_time - エージェント待機時間 : モニター・エレメント

キューに入れられたアプリケーションが、コンセントレーター構成のもとでエージェントの待機に費やした時間。値はミリ秒単位で示されます。

表 115. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 115. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 116. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

agent_wait_time モニター・エレメントを使用すると、システムがコンセントレーター環境でどの程度効率的に稼働しているかを評価するのに役立ちます。エージェント待機時間が **total_request_time** モニター・エレメントの値に対して相対的に大きい場合、要求がエージェントの待機のためにキューに入れられて長い時間を費やしていることを示します。これは、以下の 1 つ以上の状況を示している可能性があります。

- **max_coordagents** 構成パラメーターの構成が、ワークロードに対して小さすぎる。アプリケーション要求に適切なタイミングで対応するのに十分な数のコーディネーター・エージェントを使用できるようにするために、**max_coordagents**構成パラメーターの値を大きくするか、 **max_connections** 構成パラメーターに対する **max_coordagents** 構成パラメーターの比率を大きくする (両方のパラメーターとも AUTOMATIC に設定して実行している場合) 必要がある可能性があります。
- ワークロードのコミットの頻度が十分でない。コンセントレーターが効率的に処理を行うには、アプリケーションはコミットを比較的頻繁に発行して、他のアプリケーションの要求に対応するためにエージェントを解放できるようにする必要があります。アプリケーションで頻繁にコミットを行わない場合、コーディネー

ター・エージェントの数をそれに応じて多く構成し、エージェントが使用可能になるのを待機する時間を短くする必要がある可能性があります。

agent_waits_total - エージェント待機の合計 : モニター・エレメント

コンセントレーター構成において、アプリケーションが、エージェントが割り当てられるのを待機しなければならなかった回数。

表 117. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 118. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このエレメントを **agent_wait_time** モニター・エレメントと組み合わせて使用すると、コンセントレーター環境においてアプリケーション要求がエージェントを待機するのにかかる平均時間を判別できます。

agents_created_empty_pool エージェント・プールが空のために作成されたエージェント

エージェント・プールが空だったために作成されたエージェントの数。これには、DB2 を始動したときに開始したエージェントの数が含まれます (*num_initagents*)。

表 119. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法 agents_from_pool と組み合わせて使用すると、次の比率を計算できます。

エージェント・プールが空のために作成されたエージェント/プールから割り当てられたエージェント

このエレメントの使用法については、『agents_from_pool』を参照してください。

agents_from_pool - プールから割り当てられたエージェント :

エージェント・プールから割り当てられたエージェントの数。

表 120. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

このエレメントと **agents_created_empty_pool** モニター・エレメントを組み合わせて使用すると、プールが空になってエージェントの作成が必要となる頻度を判別できます。

以下の比率を使用します。

エージェント・プールが空のために作成されたエージェント/プールから割り当てられたエージェント

この比率は **num_poolagents** 構成パラメーターの適切な値を設定するために利用できます。

ほとんどのユーザーの場合、デフォルト値 100 と AUTOMATIC で最適なパフォーマンスが確保されます。

この比率は、ワークロードに応じて多少変動する可能性があります。システム上のアクティビティーが少ない場合は、追加のエージェントの作成や終了が生じる可能性があります。システム上のアクティビティーが多い場合は、エージェントの再使用が増えます。比率が低い場合は、再使用されるエージェントが多いので、システム上のアクティビティーが多いと予期されることを意味します。比率が高い場合は、再使用されるエージェントより作成されるエージェントの方が多いことを示します。問題がある場合は、**num_poolagents** 構成パラメーターの値を大きくして、比

率を低くしてください。しかし、この場合はシステム上で追加のリソースが消費されます。

agents_registered 登録済みエージェント

モニター中のデータベース・マネージャー・インスタンスに登録されているエージェント (コーディネーター・エージェントとサブエージェント) の数。

表 121. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

このエレメントは、**max_coordagents** および **max_connections** 構成パラメーターの設定や、照会内並列処理の設定を評価するときに利用してください。

agents_registered_top - エージェント最大登録数 :

データベース・マネージャーが開始されてからこれまでに同時に登録されていたエージェント (コーディネーター・エージェントとサブエージェント) の最大数。

表 122. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

このエレメントは、**max_coordagents** および **max_connections** 構成パラメーターの設定や、照会内並列処理の設定を評価するときに利用できます。

スナップショットの実行時に登録されたエージェントの数は、agents_registered モニター・エレメントにより記録されます。

agents_stolen スチールされたエージェント

データベース・マネージャーのスナップショットのレベルでは、このモニター・エレメントは、別のアプリケーション上で作業するよう再割り当てされているアプリケーションに関連したアイドル・エージェントの数を表します。アプリケーションのスナップショットのレベルでは、このモニター・エレメントは、このアプリケーション上で作業するよう再割り当てされている別のアプリケーションに関連したアイドル・エージェントの数を表します。

表 123. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

デフォルトでは **num_poolagents** 構成パラメーターは AUTOMATIC に設定されます。この場合、DB2 により自動的にアイドル・エージェントのプールが管理され、その中には別のアプリケーションに関連したアイドル・エージェントに作業を割り当てることが含まれます。

agents_top 作成されたエージェントの数

アプリケーション・レベルでは、ステートメントの実行時に使用されたエージェントの最大数です。データベース・レベルでは、これはすべてのアプリケーション用のエージェントの最大数です。

エレメント ID

agents_top

エレメント・タイプ

水準点

表 124. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	ステートメント
アプリケーション	stmt	ステートメント

使用法 照会内並列処理の実現の度合いを示します。

agents_waiting_on_token - トークン待ちエージェント :

データベース・マネージャー内でトランザクションを実行するためにトークンを待機中のエージェントの数。

注: agents_waiting_on_token モニター・エレメントは、DB2 バージョン 9.5 以降では非推奨になっています。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 125. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

このエレメントを使用すると、**maxcagents** 構成パラメーターの設定値を評価するのに役立ちます。

各アプリケーションには、データベース・マネージャー内でデータベース要求を処理するための専用コーディネーター・エージェントが 1 つずつ組み込まれます。各エージェントは、トークンを取得してから、トランザクションを実行できます。データベース・マネージャーのトランザクションを実行できるエージェントの最大数

は、 **maxcagents** 構成パラメーターの値によって制限されます。

agents_waiting_top - エージェント最大待機数 : モニター・エレメント

データベース・マネージャーが開始されてから、同時にトークンを待機していたエージェントの最大数。

注: agents_waiting_top モニター・エレメントは、DB2 バージョン 9.5 以降では非推奨になっています。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 126. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

このエレメントは、**maxcagents** 構成パラメーターの設定値を評価するときに利用できます。

スナップショットの実行時にトークンを待機していたエージェントの数は、**agents_waiting_on_token** モニター・エレメントにより記録されます。

maxcagents パラメーターをデフォルト値 (-1) に設定すると、トークンを待つエージェントがなくなるため、このモニター・エレメントの値はゼロになります。

agg_temp_tablespace_top 集約 TEMPORARY 表スペースの最上位 : モニター・エレメント

サービス・クラスでの、すべてのネスト・レベルにおける DML アクティビティーの TEMPORARY 表スペースの集約された使用量の最高水準点 (KB 単位)。 集約は、サービス・サブクラスのすべてのアクティビティーに渡る TEMPORARY 表スペースの使用量を合計して計算されます。この最高水準点は、最後のリセット以降のこの集約の最大値を表しています。 サービス・クラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合に、このモニター・エレメントは -1 を返します。このレコードのあるサブクラスと同じスーパークラスに、AGGSQLTEMPSPACE しきい値が定義され使用可能になっているサービス・サブクラスが少なくとも 1 つ必要です。そうでない場合、0 の値が返されます。

表 127. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-

使用法

このエレメントを使用して、収集された時間間隔にサービス・サブクラスのパーティションで到達した DML アクティビティーの SYSTEM TEMPORARY 表スペー

ス使用量の最大集約値を判別することができます。

aggsqltempspace_threshold_id - 集約 SQL 一時スペースしきい値 ID : モニター・エレメント

アクティビティーに適用されていた AGGSQLTEMPSPACE しきい値の数値 ID。

表 128. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、 AGGSQLTEMPSPACE しきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

aggsqltempspace_threshold_value - AggSQL 一時スペースしきい値 : モニター・エレメント

アクティビティーに適用されていた AGGSQLTEMPSPACE しきい値の上限。

表 129. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、AGGSQLTEMPSPACE しきい値がアクティビティーに適用されている場合、その値を判別します。

aggsqltempspace_threshold_violated - AggSQL 一時スペースしきい値の違反 : モニター・エレメント

オプションのモニター・エレメント。これが「Yes」に設定された場合、アクティビティーが自身に適用されていた AGGSQLTEMPSPACE しきい値に違反したことを示します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 130. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、アクティビティーに適用されていた AGGSQLTEMPSPACE しきい値にアクティビティーが違反したかどうかを判別します。

app_rqsts_completed_total - 完了したアプリケーション要求の合計 : モニター・エレメント

コーディネーターによって実行された外部 (アプリケーション) 要求の合計数。サービス・サブクラスでは、このモニター・エレメントはアプリケーション要求が完了したサブクラスについてのみ更新されます。

表 131. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 132. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このモニター・エレメントを使用して、アプリケーションからシステムにサブミットされている要求の数を判別します。

appl_con_time 接続要求開始タイム・スタンプ

アプリケーションが接続要求を開始した日時。

エレメント ID

appl_con_time

エレメント・タイプ

timestamp

表 133. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	タイム・スタンプ

使用法 このエレメントを使用すると、アプリケーションがデータベースへの接続要求を開始した日時を判別できます。

appl_id - アプリケーション ID :

アプリケーションがデータベース・マネージャーのデータベースに接続したとき、または DB2 Connect が DRDA® データベースへの接続要求を受け取ったときにこの ID が生成されます。

表 134. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	基本
DCS アプリケーション	dcs_appl_info	基本
ロック	appl_lock_list	基本

表 135. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
作業単位	-	-
接続	event_conn	-
接続	event_connheader	-
ステートメント	event_stmt	-
トランザクション[1]	event_xact	-
デッドロック[2]	event_dlconn	-
詳細付きデッドロック[2]	event_detailed_dlconn	-
アクティビティー	event_activitystmt	-
アクティビティー	event_activity	-
アクティビティー	event_activityvals	-

表 135. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR UNIT OF WORK ステートメントを使用して、トランザクション・イベントをモニターしてください。

2 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

この ID はクライアントとサーバーの両者により認識されるため、この ID を使用すると、アプリケーションのクライアント部分とサーバー部分を相関させることができます。 DB2 Connect アプリケーションでアプリケーションのクライアント部分とサーバー部分を相関させるには **outbound_appl_id** モニター・エレメントも使用する必要があります。

この ID は、ネットワーク内でユニークな ID です。アプリケーション ID にはさまざまな形式があり、データベース・マネージャー、 DB2 Connect、またはその両方を実行中のクライアントとサーバー・マシン間の通信プロトコルにより形式が異なります。どの形式の場合もピリオドで区切られた 3 つの部分で構成されます。

1. TCP/IP

   形式 `IPAddr.Port.Application instance`

   IPv4

   例

   ```
   G91A3955.F33A.02DD18143340
   ```

   詳細 IPv4 では、TCP/IP の生成するアプリケーション ID は、 3 つのセクションで構成されます。最初のセクションには IP アドレスが含まれます。 IP アドレスは、最大 8 個の 16 進文字を使用する 32 ビットの数値で表されます。 2 番目のセクションにはポート番号が含まれ、4 つの 16 進文字で表されます。 3 番目のセクションには、このアプリケーションのインスタンスのユニーク ID が含まれます。

   注: 16 進数の IP アドレスまたはポート番号が 0 から 9 で始まる場合、それぞれ G から P に変更されます。例えば、″0″ は ″G″ に、″1″ は ″H″ にそれぞれマップされます。

   IP アドレス AC10150C.NA04.006D07064947 は以下のように解釈されます。

   - IP アドレスは AC10150C のままで、これは 172.16.21.12 に変換される。

- ポート番号は NA04。最初の文字 ″N″ は ″7″ にマップされます。したがって、16 進形式のポート番号は 7A04 となり、これは 10 進形式で 31236 に変換されます。

IPv6

例

```
1111:2222:3333:4444:5555:6666:
 7777:8888.65535.0123456789AB
```

詳細 IPv6 では、TCP/IP の生成するアプリケーション ID は、3 つのセクションで構成されます。最初のセクションには、a:b:c:d:e:f:g:h 形式 (a から h はそれぞれ 4 桁の 16 進数字) の読み取り可能な 39 バイトのアドレスである IP アドレスが含まれます。 2 番目のセクションは、読み取り可能な 5 バイトのポート番号です。 3 番目のセクションは、このアプリケーションのインスタンスのユニーク・タイム・スタンプ ID です。

2. ローカル・アプリケーション

形式 `*LOCAL.DB2 instance.Application instance`

例

```
*LOCAL.DB2INST1.930131235945
```

詳細 ローカル・アプリケーション用に生成されるアプリケーション ID は、*LOCAL のストリング、DB2 インスタンスの名前、およびこのアプリケーションのインスタンスのユニーク ID の連結で構成されます。

複数データベース・パーティション・インスタンスの場合は、LOCAL は Nx に置き換えられます。 x は、クライアントからデータベースへの接続元のパーティション番号です。例えば、*N2.DB2INST1.0B5A12222841 となります。

client_protocol モニター・エレメントを使用すると、接続に使用している通信プロトコルを判別できるので、**appl_id** モニター・エレメントの形式も判別できます。

appl_id_holding_lk ロックを保持しているアプリケーション ID

このアプリケーションが取得のために待機しているオブジェクトのロックを保留しているアプリケーションのアプリケーション ID。

エレメント ID

appl_id_holding_lk

エレメント・タイプ

情報

表 136. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ロック
ロック	appl_lock_list	ロック
ロック	lock_wait	ロック

表 137. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	event_dlconn	-
詳細付きデッドロック	event_detailed_dlconn	-

使用法 このエレメントは、リソースの競合状態にあるアプリケーションの判別に利用できます。具体的には、ロックを保留しているアプリケーション・ハンドル (エージェント ID) と表 ID を識別するときに利用します。 LIST APPLICATIONS コマンドを使用してアプリケーション ID をエージェントに関連付ける情報を取得することもできます。ただし、このタイプの情報はスナップショットをとる際に収集する方が賢明です。 LIST APPLICATIONS コマンドを実行する前にアプリケーションが終了してしまうと情報を得られなくなることがあるからです。

このアプリケーションがロックの取得を待機している 1 つのオブジェクトに対して、複数のアプリケーションが共有ロックを保留していることがあるので注意してください。アプリケーションが保留しているロックのタイプについては、『lock_mode』を参照してください。アプリケーションのスナップショットをとる場合は、オブジェクトに対してロックを保留しているアプリケーション ID のうち戻されるのは 1 つだけです。ロックのスナップショットをとる場合は、オブジェクトに対してロックを保留しているすべてのアプリケーション ID が戻されます。

appl_id_oldest_xact 最も古いトランザクションを持つアプリケーション

最も古いトランザクションを持つアプリケーションのアプリケーション ID (アプリケーション・スナップショットからの *agent_id* 値に対応)。

エレメント ID

appl_id_oldest_xact

エレメント・タイプ

情報

表 138. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 このエレメントを使用すると、最も古いアクティブ・トランザクションを持つアプリケーションを判別できます。このアプリケーションにログ・スペースの解放を強制することができます。ログ・スペースを大量に消費している場合は、アプリケーションを調べて、より頻繁にコミットするよう変更できるかどうか判別してください。

ロギングを保留しているトランザクションがない場合や、最も古いトランザクションがアプリケーション ID を持たない場合 (例えば、未確定トランザクションまたは非アクティブ・トランザクション) もあります。これらの場合には、このアプリケーションの ID はデータ・ストリームに返されません。

appl_idle_time アプリケーション・アイドル時間

アプリケーションがサーバーに対して何らかの要求を出してから経過した秒数。これには、トランザクションを終了せずに、例えばコミットまたはロールバックを発行していないアプリケーションが含まれます。

エレメント ID

appl_idle_time

エレメント・タイプ

情報

表 139. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ステートメント
DCS アプリケーション	dcs_appl	ステートメント

使用法 この情報を使用すると、ユーザーが指定秒数の間アイドル状態になったときに、ユーザーに強制するようなアプリケーションをインプリメントできます。

appl_name アプリケーション名 : モニター・エレメント

クライアントで実行中のアプリケーションの名前。データベースまたは DB2 Connect サーバーが識別できる名前です。

表 140. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	基本
ロック	appl_lock_list	基本
DCS アプリケーション	dcs_appl_info	基本

表 141. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
作業単位	-	-
接続	event_connheader	-
アクティビティー	event_activity	-

使用法

このエレメントと **appl_id** を使用すると、データ項目をアプリケーションに関連付けることができます。

クライアント/サーバー環境において、この名前はデータベース接続を確立する際にクライアントからサーバーに送られます。 CLI アプリケーションは、SQLSetConnectAttr への呼び出しを使用して SQL_ATTR_INFO_PROGRAMNAME 属性を設定できます。サーバーへの接続が確立される前に

SQL_ATTR_INFO_PROGRAMNAME が設定されると、指定された値は実際のクライアント・アプリケーション名をオーバーライドし、 **appl_name** モニター・エレメント中に表示されます。

クライアント・アプリケーションのコード・ページと実行中のデータベース・システム・モニターが使用しているコード・ページが異なる場合は、 **appl_name** を変換するときに **codepage_id** を利用できます。

appl_priority アプリケーション・エージェント優先順位

このアプリケーションのために作業するエージェントの優先順位。

エレメント ID

appl_priority

エレメント・タイプ

情報

表 142. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

表 143. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	-

使用法 このエレメントを使用すると、アプリケーションが所定の優先順位で実行されているかどうかを確認できます。アプリケーションの優先順位は、管理者が設定できます。優先順位は、ガバナー・ユーティリティー (**db2gov**) で変更できます。

DB2 はガバナーを使用して、データベースに使用するアプリケーションの動作のモニターおよび変更を行います。この情報は、アプリケーションのスケジュール処理とシステム・リソースのバランス処理に使用されます。

ガバナー・デーモンはスナップショットをとることにより、アプリケーションの統計データを収集します。さらに、そのデータベース上で実行されるアプリケーションを管理する規則に照らして、この統計内容をチェックします。ガバナーが規則違反を発見すると、適切なアクションを実行します。このような規則やアクションの内容は、ガバナー構成ファイル内にユーザーが指定します。

規則に関連したアクションによりアプリケーションの優先順位が変更される場合、違反が検出されたパーティション内のエージェントの優先順位がガバナーによって変更されます。

appl_priority_type アプリケーション優先順位タイプ

アプリケーションのために作業しているエージェントの、オペレーティング・システムの優先順位タイプ。

エレメント ID

appl_priority_type

エレメント・タイプ

情報

表 144. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

表 145. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	-

使用法 使用状況に基づいて、動的優先順位がオペレーティング・システムによって再計算されます。静的優先順位は変更されません。

appl_section_inserts セクション挿入数 : モニター・エレメント

共有 SQL 作業スペースからのアプリケーションによる SQL セクション挿入数。

表 146. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

表 147. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法

実行可能セクションの作業用コピーは、共有 SQL 作業スペースに保管されます。このカウンターは、コピーが使用できなかったために挿入が必要だった場合のカウントです。

appl_section_lookups - セクション検索

共有 SQL 作業スペースからのアプリケーションによる SQL セクション検索数。

表 148. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 149. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法

個々のエージェントには、実行可能セクションの作業用コピーが保持される共有 SQL 作業スペースへのアクセス権があります。このカウンターは、アプリケーションのエージェントにより SQL 作業域がアクセスされた回数を示します。

appl_status アプリケーション状況

アプリケーションの現在の状況。

表 150. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	基本
ロック	appl_lock_list	基本

表 151. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
接続	event_conn	-

使用法

このエレメントは、潜在的なアプリケーション問題の診断に役立ちます。このフィールドの値は次に示す表でリストされています。

API 定数	説明
SQLM_AUTONOMOUS_WAIT	**自律型待機:** アプリケーションは自律型ルーチンが完了するのを待機しています。
SQLM_BACKUP	**データベースのバックアップ中:** アプリケーションはデータベースのバックアップを実行しています。
SQLM_COMMIT_ACT	**コミット・アクティブ:** 作業単位が、そのデータベース変更をコミットしています。
SQLM_COMP	**コンパイル中:** データベース・マネージャーはアプリケーションに代わって SQL ステートメントのコンパイル中またはプランのプリコンパイル中です。
SQLM_CONNECTED	**データベース接続完了:** アプリケーションがデータベース接続を開始し、要求は完了しています。
SQLM_CONNECTPEND	**データベース接続ペンディング:** アプリケーションはデータベース接続を開始していますが、要求は完了していません。

API 定数	説明
SQLM_CREATE_DB	**データベース作成中:** エージェントは、データベース作成の要求を開始しましたが、要求はまだ完了していません。
SQLM_DECOUPLED	**エージェントから分離済み:** アプリケーションに現在関連付けられているエージェントはありません。これは正常な状態です。接続コンセントレーターが使用可能なときは、専用コーディネーター・エージェントがないので、アプリケーションをコーディネーター・パーティションで分離することができます。非コンセントレーター環境では、専用コーディネーター・エージェントが常に存在するので、アプリケーションをコーディネーター・パーティションで分離することができません。
SQLM_DISCONNECTPEND	**データベース切断ペンディング:** アプリケーションはデータベースの切断を開始していますが、コマンドの実行は完了していません。アプリケーションがデータベース切断コマンドを明示的に実行していない可能性があります。切断せずにアプリケーションが終了すると、データベース・マネージャーがデータベースとの接続を切断します。
SQLM_INTR	**要求が割り込みを受けました:** 要求の割り込みが進行しています。
SQLM_IOERROR_WAIT	**表スペースを使用不可にするための待機:** アプリケーションが入出力エラーを検出し、特定の表スペースを使用不可にしようとしています。アプリケーションは、表スペースを使用不可にする前に、表スペース上のその他のすべてのアクティブなトランザクションが完了するまで待機する必要があります。
SQLM_LOAD	**データの高速ロード:** アプリケーションは、データベースにデータの「高速ロード」を実行中です。
SQLM_LOCKWAIT	**ロック待機:** 作業単位がロックを待機中です。ロックが付与されると、状況は直前の値に復帰します。
SQLM_QUIESCE_TABLESPACE	**表スペースの静止中:** アプリケーションが表スペース静止要求を実行中です。
SQLM_RECOMP	**再コンパイル中:** データベース・マネージャーがアプリケーションに代わってプランを再コンパイル (再バインド) 中です。
SQLM_REMOTE_RQST	**フェデレーテッド要求ペンディング:** アプリケーションはフェデレーテッド・データ・ソースからの結果を待っています。
SQLM_RESTART	**データベース再始動中:** アプリケーションは、クラッシュ・リカバリーを実行するためにデータベースを再始動しています。
SQLM_RESTORE	**データベースのリストア中:** アプリケーションは、バックアップ・イメージをデータベースにリストアしています。

API 定数	説明
SQLM_ROLLBACK_ACT	**ロールバック・アクティブ:** 作業単位がデータベースの変更をロールバックしています。
SQLM_ROLLBACK_TO_SAVEPOINT	**セーブポイントへのロールバック:** アプリケーションがセーブポイントにロールバック中です。
SQLM_TEND	**トランザクション終了:** 作業単位は、終了したグローバル・トランザクションの一部ですが、2 フェーズ・コミット・プロトコルの準備段階にはまだ入っていません。
SQLM_THABRT	**トランザクションをヒューリスティックにロールバック:** 作業単位は、ヒューリスティックにロールバックされたグローバル・トランザクションの一部です。
SQLM_THCOMT	**トランザクションをヒューリスティックにコミット:** 作業単位は、ヒューリスティックにコミットされたグローバル・トランザクションの一部です。
SQLM_TPREP	**トランザクション準備済み:** 作業単位は、2 フェーズ・コミット・プロトコルの準備段階に入っているグローバル・トランザクションの一部です。
SQLM_UNLOAD	**データの高速アンロード:** アプリケーションは、データベースからデータの「高速アンロード」を実行中です。
SQLM_UOWEXEC	**作業単位実行中:** データベース・マネージャーが作業単位に代わって要求を実行中です。
SQLM_UOWWAIT	**作業単位の待機中:** データベース・マネージャーがアプリケーション内の作業単位に代わって待機中です。通常、この状況はシステムがアプリケーションのコード内で実行中であることを示します。
SQLM_WAITFOR_REMOTE	**リモート要求ペンディング:** アプリケーションは、パーティション・データベース・インスタンス内のリモート・パーティションからの応答を待っています。

application_handle - アプリケーション・ハンドル : モニター・エレメント

システム全体での、アプリケーションのユニーク ID。単一パーティションのデータベースの場合は、この ID は 16 ビット・カウンターで構成されます。複数パーティションのデータベースでは、この ID はコーディネーター・パーティション番号と 16 ビット・カウンターが連結されたもので構成されます。さらに、アプリケーションが 2 次接続を行う可能性のあるパーティションには、すべて同一の ID が使用されます。

表 152. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	常に収集される

表 152. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	常に収集される
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

表 153. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	基本
ロック	appl_lock_list	基本
DCS アプリケーション	dcs_appl_info	基本

表 154. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
作業単位	-	-
接続	event_connheader	-
ステートメント	event_stmt	-
ステートメント	event_subsection	-
デッドロック[1]	event_dlconn	-
詳細付きデッドロック[1]	event_detailed_dlconn	-
しきい値違反	event_thresholdviolations	-
アクティビティー	event_activity	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

このモニター・エレメントは、**agent_id** モニター・エレメントの別名です。

appls_cur_cons - 現在接続されているアプリケーション :

現在データベースに接続されているアプリケーションの数を示します。

表 155. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
ロック	db_lock_list	基本

使用法 このエレメントを使用して、データベース内のアクティビティー・レベルおよび使用中のシステム・リソースの量を確認することができます。

maxappls および *max_coordagents* 構成パラメーターの設定値を調整するときに利用できます。例えば、この値が *maxappls* の値と常に同じ場合は、*maxappls* の値を増やすことができます。詳しくは、『*rem_cons_in*』および『*local_cons*』モニター・エレメントの項を参照してください。

appls_in_db2 - データベースで現在実行中のアプリケーション :

現在データベースに接続されており、データベース・マネージャーが要求を処理中のアプリケーションの数を示します。

表 156. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

arm_correlator アプリケーション応答測定相関関係子 : モニター・エレメント

アプリケーション応答測定 (ARM) 標準のトランザクションの ID。

表 157. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

使用法

このエレメントを使用すると、アクティビティー・イベント・モニターによって収集されるアクティビティーに関連付けられたアプリケーションもアプリケーション応答測定 (ARM) 標準をサポートする場合には、このアクティビティーをそのアプリケーションにリンクさせることができます。

associated_agents_top - 関連エージェント最大数 :

このアプリケーションに関連付けられているサブエージェントの最大数。

表 158. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

async_runstats - 非同期 RUNSTATS 要求の合計数: モニター・エレメント

データベース内のすべてのアプリケーションのリアルタイム統計収集により正常に実行された非同期 RUNSTATS アクティビティーの合計数。すべてのデータベース・パーティションで報告された値が集約されます。

表 159. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 160. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法

このエレメントを使用すると、リアルタイム統計収集により正常に実行された非同期 RUNSTATS アクティビティーの数を判別できます。この値は頻繁に変わります。システム使用量をより正確に知るには、長期にわたり特定のインターバルを設けてスナップショットを取ってください。このエレメントを **sync_runstats** および **stats_fabrications** モニター・エレメントと組み合わせて使用すると、リアルタイム統計収集に関連した異なるタイプの統計収集アクティビティーを追跡し、それらのパフォーマンスへの影響を分析する助けになります。

audit_events_total - 監査イベントの合計 : モニター・エレメント

生成された監査イベントの合計数。

表 161. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 161. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 162. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

audit_file_write_wait_time - 監査ファイル書き込み待機時間 : モニター・エレメント

監査レコードの書き込みの待機にかかった時間。値はミリ秒単位で示されます。

表 163. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 164. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE

表 164. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このモニター・エレメントを使用して、エージェントが監査イベントをオープンし、ディスクに同期的に書き込むのを待機していた時間を判別します。

典型的なシナリオでは、監査ログ・ファイルのオープンを試行するのは一度に 1 つのエージェントのみであり、他のエージェントは監査共通サブシステムにアクセスできるようになるのを待機してから、そのファイルをオープンします。そのため、この待機時間は通常、オペレーティング・システムがディスクへのファイルの書き込みを待機していた時間を表します。監査ユーティリティーは実行時に監査ログ・ファイルをロックすることがあります。その場合、エージェントが監査ログ・ファイルをオープンしてそれに書き込むのを待機する時間が通常より長くなります。非同期監査が使用可能になっている場合、非同期監査バッファーより大きい監査イベントはバッファーにではなくディスクに直接書き込まれます。この時間も、待機時間に含まれます。

特別な監査ユーティリティーを使用する場合を除いて、待機時間は、ディスクの速度と、オペレーティング・システムがどれだけの速度でディスクにデータを書き込めるかに依存します。特定のアプリケーションおよび監査構成でこの待機時間を短くするには、オペレーティング・システムをチューニングするか、より高速のディスクを使用することができます。

audit_file_writes_total - 書き込まれた監査ファイルの合計 : モニター・エレメント

監査イベントをディスクに直接書き込むためにエージェントが待機しなければならなかった回数の合計。

表 165. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 165. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 166. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このモニター・エレメントを **audit_file_write_wait_time** モニター・エレメントと組み合わせて使用して、アプリケーション要求が監査イベントをオープンし、ディスクに同期的に書き込むのを待機していた平均時間を判別します。

audit_subsystem_wait_time - 監査サブシステム待機時間 : モニター・エレメント

監査バッファー内のスペースが空くのを待機した時間。待機は、監査バッファーが満杯になり、監査デーモンがバッファーをディスクに書き込むのをエージェントが待機しなければならない場合に発生します。値はミリ秒単位で示されます。

表 167. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 167. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 168. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このモニター・エレメントを使用して、監査共通サブシステムが他のエージェントのイベント処理でビジー状態である間、監査共通サブシステムにアクセスするためにエージェントが待機する時間を判別します。

監査サブシステムの特定の共通部分には、一度に 1 つのエージェントしかアクセスできません。このモニター・エレメントの値は、監査共通サブシステムにアクセスするためにエージェントが待機しなければならない時間を示します。これには、現在の非同期バッファーを満たしたエージェントが、監査デーモンが前の非同期バッファーをディスクに書き出し終わるのを待機していた時間が含まれます。監査ログ・ファイルへの書き込みの間に待機している、または監査デーモンの要求を出すのを待機している他のエージェントも、監査共通サブシステムにアクセスしており、そこでの待機時間もこの値に反映されます。

非同期監査が使用されている場合、**audit_buf_sz** 構成パラメーターの値を変更すると、この待機時間を短くできる場合があります。 **audit_buf_sz** 構成パラメーターの値を増やしていき、それ以上値を大きくしても監査共通サブシステムの待機時間が少なくならないという値を見つけることができます。その時点では、次のバッファーが満杯になる前にデーモンが 1 つのバッファー全体をディスクに書き込むことができるほど、非同期バッファーの大きさが十分になっています。その場合、デーモンがボトルネックになることはなくなります。 **audit_buf_sz** 構成パラメーターをそのような値にまで増やすと、システム障害が起きた場合に失われる可能性がある監査レコードの数が多すぎるという場合には、オペレーティング・システムをチューニングするか、より高速なディスクを使用することによって、待機時間を短くする

ことができます。待機時間をさらに短くする必要がある場合は、監査ポリシーを使用して、生成される監査イベントの数を減らしてください。

audit_subsystem_waits_total - 監査サブシステム待機の合計 : モニター・エレメント

監査がバッファーの書き込みを待機した回数。

表 169. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 170. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このモニター・エレメントを使用して、エージェントが監査共通サブシステムにアクセスしなければならなかった合計回数を判別します。 1 つの監査イベントを生成する場合、イベントを記録するために、監査共通サブシステムへのアクセスが必要ない場合もあれば、1 回のアクセスが必要な場合、あるいは複数回のアクセスが必要な場合もあります。生成された監査イベントの正確な数を判別するには、**audit_events_total** モニター・エレメントを使用します。

auth_id 許可 ID

モニターされているアプリケーションを呼び出したユーザーの許可 ID。この ID は、DB2 Connect のゲートウェイ・ノード上では、ホスト上にあるユーザーの許可 ID となります。

表 171. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	基本
ロック	appl_lock_list	基本
DCS アプリケーション	dcs_appl_info	基本

表 172. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
作業単位	-	-
接続	event_connheader	-

使用法

明示的なトラステッド接続では、ユーザーを切り替えた直後に **auth_id** 値が変更されるわけではありません。直後ではなく、ユーザーを切り替えてから最初にデータベースにアクセスしたときに、**auth_id** が更新されます。これは、ユーザー切り替え操作には必ずその後の操作が続くためです。

このエレメントを使用すると、アプリケーションを呼び出したユーザーを判別できます。

authority_bitmap ユーザー許可レベル : モニター・エレメント

ユーザー、およびユーザーが所属するグループに付与された権限。これには、ユーザーに付与されたロールに付与された権限、およびユーザーが所属するグループに付与されたロールに付与された権限も含まれます。ユーザーに付与された権限、またはユーザーに付与されたロールに付与された権限は、ユーザー権限と見なされます。ユーザーが所属するグループに付与された権限、またはユーザーが所属するグループに付与されたロールに付与された権限は、グループ権限と見なされます。

表 173. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本
アプリケーション	appl_info	基本

表 174. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	-

使用法

authority_bitmap モニター・エレメントには、配列の形式があります。各配列エレメントは、ユーザー ID に特定の権限が付与されているかどうか、およびユーザーがどのようにその権限を受け取ったかを示す単一文字です。

個々の配列エレメントは、sql.h ファイルで定義された索引値によって索引付けされています。 authority_bitmap 配列の索引の値は、*権限索引*と呼ばれています。例えば、SQL_DBAUTH_SYSADM は、ユーザーに SYSADM 権限があるかを判別するための索引です。

権限索引で識別される authority_bitmap 配列にある 1 つのエレメントの値は、権限が許可 ID により保持されているかを示します。許可 ID が保持される方法を判別するには、権限索引により識別される配列エレメントごとに、sql.h の以下の定義を使用してください。

SQL_AUTH_ORIGIN_USER
このビットがオンである場合、許可 ID には、ユーザーに付与された権限、またはユーザーに付与されたロールに付与された権限があります。

SQL_AUTH_ORIGIN_GROUP
このビットがオンである場合、許可 ID には、グループに付与された権限、またはグループに付与されたロールに付与された権限があります。

例えば、ユーザーが DBADM 権限を保持しているかを判別するには、以下の値を確認します。

```
authority_bitmap[SQL_DBAUTH_DBADM]
```

DBADM 権限がユーザーにより直接保持されているかを判別するには、以下を確認します。

```
authority_bitmap[SQL_DBAUTH_DBADM] & SQL_AUTH_ORIGIN_USER
```

authority_lvl ユーザー許可レベル : モニター・エレメント

アプリケーションに対して付与されている最高権限レベル。

注: authority_lvl モニター・エレメントは、DB2 データベース・バージョン 9.5 以降では推奨されていません。その代わりに、authority_bitmap モニター・エレメントを使用してください。 395 ページの『authority_bitmap ユーザー許可レベル : モニター・エレメント』を参照してください。

表 175. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本
アプリケーション	appl_info	基本

表 176. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	-

使用法 アプリケーションで許可される操作は、直接または間接的に付与されます。

sql.h の次の定義を使用して、ユーザーに明示的に付与されている許可を判別できます。

- SQL_SYSADM
- SQL_DBADM
- SQL_CREATETAB
- SQL_BINDADD
- SQL_CONNECT
- SQL_CREATE_EXT_RT
- SQL_CREATE_NOT_FENC
- SQL_SYSCTRL
- SQL_SYSMAINT

sql.h の次の定義を使用して、グループまたは PUBLIC 権限から継承されている間接許可を判別できます。

- SQL_SYSADM_GRP
- SQL_DBADM_GRP
- SQL_CREATETAB_GRP
- SQL_BINDADD_GRP
- SQL_CONNECT_GRP
- SQL_CREATE_EXT_RT_GRP
- SQL_CREATE_NOT_FENC_GRP
- SQL_SYSCTRL_GRP
- SQL_SYSMAINT_GRP

auto_storage_hybrid - ハイブリッド自動ストレージの表スペース標識 : モニター・エレメント

表スペースが非自動ストレージ・コンテナーを含む自動ストレージ表スペースである場合、このモニター・エレメントは値 1 を戻します。それ以外の場合は、値 0 を戻します。

表 177. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

ハイブリッド自動ストレージ表スペースは、ALTER TABLESPACE コマンドを使用して自動ストレージによって管理するように変換されているものの、まだリバランスされていない表スペースです。この表スペースには、依然として非自動ストレージ・コンテナーが含まれます。リバランスされた後、表スペースには自動ストレージ・コンテナーのみが含まれ、ハイブリッド表スペースと見なされなくなります。

automatic - バッファー・プール自動 : モニター・エレメント

特定のバッファー・プールでセルフチューニングが使用可能になっているかどうかを示します。このエレメントは、バッファー・プールでセルフチューニングが使用可能になっている場合は 1 に、使用可能になっていない場合は 0 に設定されます。

表 178. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE

bin_id ヒストグラム・ビン ID : モニター・エレメント

ヒストグラム・ビンの ID。**bin_id** はヒストグラム内で固有です。

表 179. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_histogrambin	-

使用法

このエレメントを使用すると、同じヒストグラム内でビンを区別できます。

binds_precompiles 試行されたバインド/プリコンパイル

試みられたバインドおよびプリコンパイルの数。

エレメント ID
binds_precompiles

エレメント・タイプ
カウンター

表 180. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 181. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 このエレメントを使用すると、データベース・マネージャー内の現在のアクティビティーのレベルがわかります。

この値には *int_auto_rebinds* のカウントは含まれませんが、 REBIND PACKAGE コマンドの結果として起こるバインド数は含まれます。

block_ios - ブロック入出力要求数 : モニター・エレメント

ブロック入出力要求の数。より厳密には、 DB2 がバッファー・プールのブロック域に対してページの順次プリフェッチを実行する回数。

表 182. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 183. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	bufferpool	バッファー・プール

使用法

ブロック・ベースのバッファー・プールを使用可能にすると、このモニター・エレメントがブロック入出力の実行頻度を報告します。それ以外の場合、このモニター・エレメントは 0 を戻します。ブロック入出力要求の数がモニターされるのは、ブロック・ベースのバッファー・プールの使用中に順次プリフェッチが実行される間だけです。

ブロック・ベースのバッファー・プールを使用可能にしてあり、この数が非常に小さい場合、またはベクトル化入出力の数 (**vectored_ios** モニター・エレメントの値) に近い場合は、ブロック・サイズを変更することを考慮してください。このようなときは、以下の状態を示している場合があります。

- バッファー・プールにバインドされている 1 つ以上の表スペースのエクステント・サイズが、バッファー・プールに指定されているブロック・サイズよりも小さい。
- プリフェッチ要求されたページのうち、いくつかのページがバッファー・プールのページ・エリアにすでに存在している。

プリフェッチャーはそれぞれのバッファー・プール・ブロックに無駄なページが多少あっても見過ごしますが、無駄なページの数が非常に多くなると、プリフェッチャーはバッファー・プールのページ・エリアにベクトル化入出力を実行します。

ブロック・ベースのバッファー・プールを使用することで順次プリフェッチのパフォーマンスが改善される点を活用するには、ブロック・サイズに適切な値を選ぶことが非常に重要です。しかし、エクステント・サイズの異なる複数の表スペースが同じブロック・ベースのバッファー・プールにバインドされていることがあるので、値の選択が難しいこともあります。最適なパフォーマンスを得るためには、同

じエクステント・サイズの表スペースを、ブロック・サイズがエクステント・サイズと等しいブロック・ベースのバッファー・プールにバインドすることをお勧めします。表スペースのエクステント・サイズがブロック・サイズより大きい場合にも良好なパフォーマンスが得られますが、ブロック・サイズよりも小さい場合にはパフォーマンスが低下します。

例えば、エクステント・サイズが 2 でブロック・サイズが 8 の場合は、ブロック入出力の代わりにベクトル化入出力が使用されます (ブロック入出力では 6 ページの無駄が発生します)。ブロック・サイズを 2 に下げると、この問題は解決できます。

blocking_cursor ブロック・カーソル

このエレメントは、実行中のステートメントがブロッキング・カーソルを使用しているかどうかを示します。

エレメント ID

blocking_cursor

エレメント・タイプ

情報

表 184. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント
DCS ステートメント	dcs_stmt	ステートメント

表 185. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
詳細付きデッドロック	event_detailed_dlconn	-
ステートメント	event_stmt	-

使用法 照会のデータ転送にブロッキングを使用すると、パフォーマンスが改善されます。照会に使用される SQL はブロッキングの使用と関係があるため、多少の変更が必要になる場合があります。

blocks_pending_cleanup クリーンアップ保留中のロールアウト済みブロック : モニター・エレメント

ロールアウト削除に続いて非同期クリーンアップを保留中のデータベースにおける MDC 表ブロックの合計数。

表 186. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	-
データベース	event_db	-

使用法

このエレメントを使用すると、延期できるクリーンアップ・ロールアウトの削除の後で使用可能なストレージとしてシステムに解放されていない MDC 表ブロックの数を判別できます。

bottom ヒストグラム・ビンの最下位 : モニター・エレメント

ヒストグラム・ビンの範囲外の最終点。このモニター・エレメントの値は、前のヒストグラム・ビンがある場合には、その最終範囲に含まれる最初の値になります。

表 187. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_histogrambin	-

使用法

このエレメントと対応する **top** エレメントを一緒に使用して、ヒストグラム中のビンの範囲を判別します。

boundary_leaf_node_splits - 境界リーフ・ノードの分割 : モニター・エレメント

挿入操作時に境界リーフ・ノードが分割された回数。

表 188. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

bp_cur_buffsz バッファー・プールの現行サイズ

現行のバッファー・プール・サイズ。

表 189. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	bufferpool_nodeinfo	バッファー・プール

bp_id バッファー・プール ID : モニター・エレメント

このエレメントには、モニターされているバッファー・プールのバッファー・プール ID が含まれます。

表 190. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	バッファー・プール	基本

bp_name - バッファー・プール名 : モニター・エレメント

バッファー・プールの名前。

表 191. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE

表 192. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	bufferpool	基本

使用法 各データベースには少なくとも 1 つのバッファー・プールが必要です。必要に応じて、単一のデータベースのためにそれぞれサイズの違ういくつかのバッファー・プールを作成できます。 CREATE、ALTER、および DROP BUFFERPOOL ステートメントを使用して、バッファー・プールを作成、変更、ドロップすることが可能です。

データベースを作成する場合、データベースには IBMDEFAULTBP というデフォルトのバッファー・プールが設定され、そのサイズはプラットフォームによって決まります。さらに、それぞれ異なるページ・サイズに対応する次のようなシステム・バッファー・プールも設定されます。

- IBMSYSTEMBP4K
- IBMSYSTEMBP8K
- IBMSYSTEMBP16K
- IBMSYSTEMBP32K

これらのシステム・バッファー・プールは変更できません。

bp_new_buffsz 新規バッファー・プール・サイズ

データベースが再始動されるとバッファー・プールが変更されるサイズ。 ALTER BUFFERPOOL ステートメントが DEFERRED として実行されると、データベースを停止するか再始動するまでバッファー・プール・サイズは変更されません。

表 193. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	bufferpool_nodeinfo	バッファー・プール

bp_pages_left_to_remove 除去残ページ数

バッファー・プールのサイズ変更が完了する前に、バッファー・プールからの除去が残っているページ数。これは IMMEDIATE として実行される ALTER BUFFERPOOL ステートメントによって呼び出されるバッファー・プール・サイズ変更操作にのみ適用されます。

表 194. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	bufferpool_nodeinfo	バッファー・プール

bp_tbsp_use_count バッファー・プールにマップされている表スペースの数

このバッファー・プールを使用している表スペースの数。

表 195. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	bufferpool_nodeinfo	バッファー・プール

buff_free 現在空いている FCM バッファー

このエレメントは、現在空いている FCM バッファーの数を示します。

エレメント ID
buff_free

エレメント・タイプ
ゲージ

表 196. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	fcm	基本

使用法 現在空いている FCM バッファーの数を *fcm_num_buffers* 構成パラメーターとともに使用して、現在の FCM バッファー・プール使用率を判別できます。この情報を使用すると、*fcm_num_buffers* を調整できます。

buff_free_bottom 空き FCM バッファーの最小数

処理中に到達した空き FCM バッファーの最小数。

表 197. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	fcm	基本

使用法 このエレメントと *fcm_num_buffers* 構成パラメーターを組み合わせて使用すると、 FCM バッファー・プール使用率の最大値を判別できます。buff_free_bottom が小さい場合は、*fcm_num_buffers* を大きくして、処理中に FCM バッファーが不足しないようにしてください。 buff_free_bottom が大きい場合は、*fcm_num_buffers* を小さくすると、システム・リソースを節約できます。

byte_order イベント・データのバイト・オーダー

数値データのバイト配列。具体的には、イベント・データ・ストリームが「ビッグ・エンディアン」サーバー (RS/6000® など) で生成されたか、あるいは「リトル・エンディアン」サーバー (Windows 2000 が稼働する Intel 系 PC) で生成されたかを示します。

表 198. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
イベント・ログ・ヘッダー	event_log_header	-

使用法 「ビッグ・エンディアン」サーバーの整数のバイト・オーダーは、「リトル・エンディアン」サーバーのバイト・オーダーとは逆になるため、データ・ストリームの数値データを解釈するときにこの情報が必要になります。

データを処理するアプリケーションが一方のタイプのコンピューター・ハードウェア上で実行していることを認識し (例えば、ビッグ・エンディアン・コンピューター)、イベント・データがもう一方のタイプのコンピューター・ハードウェア上で生成された場合 (例えば、リトル・エンディアン・コンピューター)、モニター・アプリケーションはこれらのデータを解釈する前に数値データ・フィールドのバイトを逆転する必要があります。タイプが同じ場合は、バイトの再配列は必要ありません。

このエレメントは、次のいずれかの API 定数に設定できます。

- SQLM_BIG_ENDIAN
- SQLM_LITTLE_ENDIAN

cat_cache_inserts - カタログ・キャッシュ挿入数

システムが表記述子または許可情報をカタログ・キャッシュに挿入しようとした回数。

エレメント ID

cat_cache_inserts

エレメント・タイプ

カウンター

表 199. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 200. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 カタログ・キャッシュ検索と組み合わせると、次の公式を使用してカタログ・キャッシュ・ヒット率を計算できます。

```
1 - (カタログ・キャッシュ挿入数 / カタログ・キャッシュ検索数)
```

このエレメントの使用法については、『cat_cache_lookups』を参照してください。

cat_cache_lookups カタログ・キャッシュ検索

表の記述子情報または許可情報を取得するためにカタログ・キャッシュが参照された回数。

エレメント ID

cat_cache_lookups

エレメント・タイプ

カウンター

表 201. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 202. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 このエレメントには、カタログ・キャッシュへの正常に行われたアクセスと失敗したアクセスの両方が含まれます。カタログ・キャッシュは、次の場合に参照されます。

- SQL ステートメントのコンパイル中に、表、ビュー、または別名を処理したとき。
- データベース許可情報にアクセスがあったとき。
- SQL ステートメントのコンパイル中にルーチンを処理したとき。

カタログ・キャッシュ・ヒット率の計算には次の数式を使用します。

```
(1 - (cat_cache_inserts / cat_cache_lookups))
```

この値は、カタログ・キャッシュがどの程度カタログ・アクセスを回避しているかを示します。この比率が高い場合 (0.8 を超える値)、キャッシュは効果的に動作しています。比率が低い場合は、 *catalogcache_sz* を大きくする必要があることを示します。データベースへの最初の接続の直後は、この比率は高くなります。

表、ビュー、別名などに関係するデータ定義言語 (DDL) SQL ステートメントは、そのようなオブジェクトに関する表記述子情報をカタログ・キャッシ

ュから取り除くため、それらのオブジェクトは次の参照で再挿入されることになります。さらに、データベース許可およびルーチンの実行特権のための GRANT および REVOKE のステートメントによって、該当する許可情報がカタログ・キャッシュから取り除かれます。したがって、DDL ステートメントと GRANT/REVOKE ステートメントを多用した場合も、この比率は大きくなります。

「カタログ・キャッシュ・サイズ」構成パラメーターについて詳しくは、「*管理ガイド*」を参照してください。

cat_cache_overflows カタログ・キャッシュ・オーバーフロー数

割り振られたメモリーの境界からカタログ・キャッシュがオーバーフローした回数。

表 203. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 204. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法

このエレメントと **cat_cache_size_top** モニター・エレメントを組み合わせて使用すると、オーバーフローを回避するのにカタログ・キャッシュのサイズを大きくする必要があるかどうかを判別できます。

カタログ・キャッシュのスペースは、どのトランザクションでも現在使用されていない、表、ビュー、または別名に関する表記述子情報や、許可情報を除去することで取り戻します。

cat_cache_overflows モニター・エレメントの値が大きい場合は、ワークロードに対してカタログ・キャッシュが小さすぎることが考えられます。カタログ・キャッシュを大きくすると、パフォーマンスが改善されることがあります。多数の表、ビュー、別名、ユーザー定義関数またはストアード・プロシージャーを参照する多数の SQL ステートメントを、 1 つの作業単位にコンパイルするトランザクションがワークロードに含まれている場合は、 1 つのトランザクションにコンパイルする SQL ステートメントの数を少なくすると、カタログ・キャッシュのパフォーマンスが改善されることがあります。あるいは多数の表、ビュー、別名、ユーザー定義関数またはストアード・プロシージャーを参照する多数の SQL ステートメントが入ったパッケージのバインドがワークロードに含まれる場合は、パッケージを分割してその中に含まれる SQL ステートメントの数を少なくすると、パフォーマンスが改善されることがあります。

cat_cache_size_top - カタログ・キャッシュの最高水準点 : モニター・エレメント

カタログ・キャッシュが到達した最大論理サイズ。

表 205. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 206. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法

このエレメントは、データベースが活動化されて以降、データベースでワークロードを実行したときに論理的に必要となったカタログ・キャッシュの最大バイト数を示します。

カタログ・キャッシュは論理サイズで管理されます。論理サイズにはメモリー管理のオーバーヘッドが含まれません。データベース・スナップショット内の **pool_watermark** エレメントは、カタログ・キャッシュによって使用されたメモリーの物理最高水準点値を示します。カタログ・キャッシュのモニター作業とチューニング作業には、物理サイズよりも論理サイズを使用するようにしてください。

カタログ・キャッシュがオーバーフローした場合、このエレメントは、オーバーフロー時にカタログ・キャッシュが到達した最大サイズになります。このような状態が発生したかどうかを確認するには、**cat_cache_overflows** モニター・エレメントをチェックしてください。

ワークロードに必要なカタログ・キャッシュの最小サイズは次のように決定できます。

```
maximum catalog cache size / 4096
```

この結果を切り上げた整数が、オーバーフローを避けるためにカタログ・キャッシュが必要とする 4K ページの最小数になります。

catalog_node カタログ・ノード番号

データベース・カタログ表が保管されているノードのノード番号。

エレメント ID

catalog_node

エレメント・タイプ

情報

表 207. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 208. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 カタログ・ノードは、すべてのシステム・カタログ表が保管されているノードです。システム・カタログ表へのアクセスは、すべてこのノードを通る必要があります。

catalog_node_name カタログ・ノード・ネットワーク名

カタログ・ノードのネットワーク名。

エレメント ID

catalog_node_name

エレメント・タイプ

情報

表 209. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 210. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 このエレメントを使用して、データベースのロケーションを判別できます。

ch_free 現在空いているチャネル

このエレメントは、現在空いているノード間通信チャネルの数を示します。

表 211. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	fcm	基本

使用法 通信チャネルの現在の空き数と *fcm_num_channels* 構成パラメーターを組み合わせて使用すると、現在の接続項目使用率を判別できます。この情報を使用すると、*fcm_num_channels* を調整できます。

ch_free_bottom 空いているチャネルの最小

処理中に到達したノード間空き通信チャネルの最小数。

表 212. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	fcm	基本

使用法 このエレメントと *fcm_num_channels* 構成パラメーターを組み合わせて使用すると、接続項目使用率の最大値を判別できます。

client_acctng - クライアント・アカウンティング・ストリング : モニター・エレメント

この接続で sqleseti API が発行された場合に、ロギングおよび診断の目的でターゲット・データベースに渡されたデータです。この接続、作業単位、またはアクティビティーの CLIENT_ACCTNG 特殊レジスターの現行値。

このモニター・エレメントは、**tpmon_acc_str** モニター・エレメントと同義です。**client_acctng** モニター・エレメントは、DB2 バージョン 9.7 で導入された未フォーマットの表に書き込む表関数およびイベント・モニターのモニターに使用されます。**tpmon_acc_str** モニター・エレメントは、表、ファイル、およびパイプに書き込むスナップショット・モニターおよびイベント・モニターに使用されます。

表 213. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	常に収集される
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	常に収集される
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

表 214. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロック	-	-
作業単位	-	-

使用法

このエレメントは、問題判別およびアカウンティングの目的で使用します。

client_applname - クライアント・アプリケーション名 : モニター・エレメント

この接続で sqleseti API が発行された場合に、トランザクションを実行中のサーバー・トランザクション・プログラムを示します。この接続、作業単位、またはアクティビティーの CLIENT_APPLNAME 特殊レジスターの現行値。

このモニター・エレメントは、**tpmon_client_app** モニター・エレメントと同義です。 **client_applname** モニター・エレメントは、DB2 バージョン 9.7 で導入された未フォーマットの表に書き込む表関数およびイベント・モニターのモニターに使用されます。**tpmon_client_app** モニター・エレメントは、表、ファイル、およびパイプに書き込むスナップショット・モニターおよびイベント・モニターに使用されます。

表 215. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	常に収集される
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	常に収集される

表 216. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロック	-	-
作業単位	-	-

使用法

このエレメントは、問題判別およびアカウンティングの目的で使用します。

client_db_alias アプリケーションで使用するデータベース別名

アプリケーションがデータベースに接続するときに指定するデータベースの別名。

エレメント ID

client_db_alias

エレメント・タイプ

情報

表 217. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	基本
ロック	appl_lock_list	基本

表 218. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_connheader	-

使用法 このエレメントを使用して、アプリケーションがアクセスしている実際のデータベースを識別できます。この名前と *db_name* 間のマッピングには、クライアント・ノードとデータベース・マネージャーのサーバー・ノードにあるデータベース・ディレクトリーを使用します。

この別名は、データベース接続要求を発行したデータベース・マネージャー内に定義されている別名です。

データベースの別名が異なると認証タイプが異なるので、このエレメントを認証タイプの判別に利用することもできます。

client_idle_wait_time - クライアントのアイドル待機時間 : モニター・エレメント

このモニター・エレメントは、クライアントによる次の要求の送信を待機していた時間を記録します。値はミリ秒単位で示されます。

表 219. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 220. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このモニター・エレメントを使用して、クライアントからの要求の待機にではなく、要求の処理にかかった時間を判別します。クライアントのアイドル時間が大きい場合、サーバー上ではなくクライアント上で対処する必要のあるパフォーマンス上の問題を示している可能性があります。

client_pid クライアント・プロセス ID

データベースへの接続を行ったクライアント・アプリケーションのプロセス ID。

表 221. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
アプリケーション	appl	基本
DCS アプリケーション	dcs_appl_info	基本

表 222. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	-
接続	event_connheader	-

使用法

このエレメントを使用して、CPU および入出力時間などのモニター情報をクライアント・アプリケーションに関連付けることができます。

DRDA AS 接続のとき、このエレメントは 0 に設定されます。

client_platform クライアント・オペレーティング・プラットフォーム

クライアント・アプリケーションが稼働中のオペレーティング・システム。

表 223. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
アプリケーション	appl	基本
DCS アプリケーション	dcs_appl_info	基本

表 224. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	-
接続	event_connheader	-

使用法

このエレメントを使用して、リモート・アプリケーションの問題判別を行えます。このフィールドの値は、ヘッダー・ファイルの sqlmon.h にあります。

client_prdid クライアント製品およびバージョン ID : モニター・エレメント

クライアント上で実行中の製品とバージョン。

表 225. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	基本
DCS アプリケーション	dcs_appl_info	基本

表 226. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	-
接続	event_connheader	-

使用法

このエレメントを使用すると、 IBM® データ・サーバー・クライアントの製品とコード・バージョンを識別できます。 PPPVVRRM の形式になっています。各部分の定義は次のとおりです。

- PPP は製品を示し、 DB2 製品の場合は 『SQL』 である。
- VV は 2 桁でバージョン番号を示す (バージョン番号が 1 桁の場合には、高位の桁は 0 になります)。
- RR は 2 桁でリリース番号を示す (リリース番号が 1 桁の場合には高位の桁は 0 になります)。
- M は 1 文字で修正レベルを示します (0-9 または A-Z)。

client_protocol クライアント通信プロトコル

クライアント・アプリケーションがサーバーとの通信に使用している通信プロトコル。

表 227. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
アプリケーション	appl	基本
DCS アプリケーション	dcs_appl_info	基本

表 228. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	-
接続	event_connheader	-

使用法

このエレメントを使用して、リモート・アプリケーションの問題判別を行えます。このフィールドの値は以下のとおりです。

SQLM_PROT_UNKNOWN
クライアントは、不明のプロトコルを使用して通信を行っています。この値は、今後のクライアントが以前のレベルのサーバーに接続した場合にのみ戻されます。

SQLM_PROT_LOCAL
クライアントは、サーバーと同一のノードで実行されており、使用中の通信プロトコルはありません。

SQLM_PROT_TCPIP
TCP/IP

client_userid - クライアントのユーザー ID : モニター・エレメント

sqleseti API が使用された場合は、トランザクション・マネージャーが生成してサーバーに提供したクライアント・ユーザー ID。この接続、作業単位、またはアクティビティーの CLIENT_USERID 特殊レジスターの現行値。

このモニター・エレメントは、**tpmon_client_userid** モニター・エレメントと同義です。 **client_userid** モニター・エレメントは、DB2 バージョン 9.7 で導入された未フォーマットの表に書き込む表関数およびイベント・モニターのモニターに使用されます。**tpmon_client_userid** モニター・エレメントは、表、ファイル、およびパイプに書き込むスナップショット・モニターおよびイベント・モニターに使用されます。

表 229. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	常に収集される
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

表 229. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	常に収集される
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

表 230. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロック	-	-
作業単位	-	-

使用法

アプリケーション・サーバーまたはトランザクション処理モニター環境でこのエレメントを使用すると、トランザクションの実行対象になっているエンド・ユーザーを識別できます。

client_wrkstnname - クライアント・ワークステーション名 : モニター・エレメント

この接続で sqleseti API が発行された場合に、クライアントのシステムまたはワークステーション (CICS EITERMID など) を示します。この接続、作業単位、またはアクティビティーの CLIENT_WRKSTNNAME 特殊レジスターの現行値。

このモニター・エレメントは、**tpmon_client_wkstn** モニター・エレメントと同義です。 **client_wrkstnname** モニター・エレメントは、DB2 バージョン 9.7 で導入された未フォーマットの表に書き込む表関数およびイベント・モニターのモニターに使用されます。**tpmon_client_wkstn** モニター・エレメントは、表、ファイル、およびパイプに書き込むスナップショット・モニターおよびイベント・モニターに使用されます。

表 231. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	常に収集される
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	常に収集される

表 231. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

表 232. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロック	-	-
作業単位	-	-

使用法

このエレメントを使用すると、ノード ID、端末 ID、またはその他の同様の ID を使用して、ユーザーのマシンを識別できます。

codepage_id アプリケーションで使用するコード・ページ ID

コード・ページ ID。

表 233. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	基本
ロック	appl_lock_list	基本
DCS アプリケーション	dcs_appl_info	基本

表 234. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
イベント・ログ・ヘッダー	event_log_header	-
接続	event_connheader	-

使用法 スナップショット・モニター・データの場合は、モニター対象のアプリケーションが開始されたパーティションでのコード・ページとなります。この ID は、リモート・アプリケーションの問題判別に使用できます。この情報を使用すると、アプリケーションのコード・ページとデータベースのコード・ページ (DRDA ホスト・データベースの場合はホスト CCSID) の間でデータ変換がサポートされるように指定できます。サポート対象のコード・ページについては、「*管理ガイド*」を参照してください。

イベント・モニター・データの場合は、イベント・データを収集したデータベースのコード・ページとなります。このエレメントを使用すると、使用中のイベント・モニター・アプリケーションの実行に使用しているコード・ページとデータベースが使用しているコード・ページが異なるコード・ページかどうかを判別できます。イベント・モニターが書き込むデータには、デー

タベースのコード・ページが使用されます。使用しているイベント・モニター・アプリケーションが別のコード・ページを使用する場合は、データを読み取るのに文字変換が必要になる場合があります。

comm_private_mem - コミット済み専用メモリー :

スナップショット時点で、データベース・マネージャーのインスタンスが現在コミットしている専用メモリーの量。返される comm_private_mem 値は、Windows オペレーティング・システムのみに関係します。

表 235. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

commit_sql_stmts 試行されたコミット・ステートメント

試行された SQL COMMIT ステートメントの合計数。

表 236. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
データベース	dbase_remote	基本
アプリケーション	appl	基本
アプリケーション	appl_remote	基本
DCS データベース	dcs_dbase	基本
DCS アプリケーション	dcs_appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 237. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 モニター期間中にこのカウンターの変化量が少ない場合は、各アプリケーションのコミット頻度が少ないことを示し、ロギングとデータの並行性について問題となる場合があります。

このエレメントを使用すると、次の項目を合計して合計作業単位数も計算できます。

```
   commit_sql_stmts
 + int_commits
 + rollback_sql_stmts
 + int_rollbacks
```

注: 計算した作業単位に含まれるのは、次の時点以降の作業単位だけです。

- データベースへの接続 (データベース・レベルの情報の場合は、最初の接続時刻)
- データベース・モニター・カウンターの最後のリセット

この計算は、データベース・レベルとアプリケーション・レベルのいずれでも行えます。

comp_env_desc コンパイル環境 : モニター・エレメント

このエレメントは、SQL ステートメントをコンパイルする際に使用されるコンパイル環境に関する情報を保管します。

表 238. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 239. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
詳細付きデッドロック履歴値	event_stmt_history	-
詳細付きデッドロック履歴	event_stmt_history	-
アクティビティー	event_activitystmt	-

使用法 このモニター・エレメントはコンパイル環境記述をバイナリー・ラージ・オブジェクトに保管します。 この情報を読み取り可能な形式で表示するには、COMPILATION_ENV 表関数を使用します。

このエレメントを COMPILATION_ENV 表関数への入力として、または SET COMPILATION ENVIRONMENT SQL ステートメントへの入力として提供することができます。

completion_status 完了状況 : モニター・エレメント

作業単位の状況。

表 240. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	-

使用法

このエレメントを使用して、作業単位が終了した原因がデッドロックによるものか、または異常終了によるものかを判別できます。 sqllib/misc/DB2EvmonUOW.xsd ファイルに可能な値がリストされています。

- UNKNOWN
- COMMIT
- ROLLBACK

- GLOBAL_COMMIT
- GLOBAL_ROLLBACK
- XA_END
- XA_PREPARE

con_elapsed_time 最新の接続経過時間

このホスト・データベースから最後に切断された DCS アプリケーションが接続されていた経過時間。

エレメント ID

con_elapsed_time

エレメント・タイプ

時間

表 241. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	タイム・スタンプ

使用法 アプリケーションがホスト・データベースへの接続を維持していた時間の長さの標識として、このエレメントを使用します。

con_local_dbases 現行接続を持つローカル・データベース

アプリケーションが接続されているローカル・データベースの数。

表 242. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法 この数値は、データベース・レベルでデータを収集する際に、予想されるデータベース情報レコードの数を示します。

アプリケーションは、ローカルまたはリモートで実行中ですが、データベース・マネージャー内で作業単位を実行している場合としていない場合があります。

con_response_time 接続の最新応答時間

このデータベースに最後に接続された DCS アプリケーションにおける、接続処理の開始と実際の接続の確立との間の経過時間。

エレメント ID

con_response_time

エレメント・タイプ

時間

表 243. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	タイム・スタンプ

使用法 アプリケーションが特定のホスト・データベースに接続するために現在かかっている時間の標識として、このエレメントを使用します。

concurrent_act_top 並行アクティビティーの最上位 : モニター・エレメント

最後にリセットされてからのサービス・サブクラスにおける並行アクティビティー (すべてのネスト・レベル) の最高水準点。

表 244. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-

使用法

このエレメントを使用して、収集された時間間隔にサービス・サブクラス用のパーティションで到達したアクティビティー (ネストされたアクティビティーを含む) の並行性の最大数を調べることができます。

concurrent_connection_top 並行接続の最上位 : モニター・エレメント

最後にリセットされてからのこのサービス・クラスにおける並行コーディネーター接続の最高水準点。同じスーパークラスを持つすべてのサブクラスにおいて、このフィールドの値は同じです。

表 245. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-

使用法

このエレメントは、現在の最高水準点がある場所を示すことにより、接続並行性のどの位置にしきい値を設定するかを判別する上で役立つ場合があります。さらに、そのようなしきい値が正しく構成され、作動しているかを検証する上でも役立ちます。

concurrent_wlo_act_top 並行 WLO アクティビティーの最上位 : モニター・エレメント

最後にリセットされてからの、このワークロードの任意のオカレンスにおける並行アクティビティー (すべてのネスト・レベル) の最高水準点。

表 246. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats	-

使用法

このエレメントを使用して、収集された時間間隔にこのワークロードの任意のオカレンス用のパーティションで到達した並行アクティビティーの最大数を調べることができます。

concurrent_wlo_top 並行ワークロード・オカレンスの最上位 : モニター・エレメント

最後にリセットされてからのワークロードの並行オカレンスの最高水準点。

表 247. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats	-
統計	event_scstats	-

使用法

このエレメントを使用して、収集された時間間隔にワークロード用のパーティションで到達したワークロード・オカレンスの並行性の最大数を調べることができます。

concurrentdbcoordactivities_db_threshold_id 並行データベース・コーディネーター・アクティビティーのデータベースしきい値 ID : モニター・エレメント

アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES データベースしきい値の ID。

表 248. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES データベースしきい値がアクティビティーに適用されていた場合、どのデータベースしきい値が適用されていたかを判別します。

concurrentdbcoordactivities_db_threshold_queued 並行データベース・コーディネーター・アクティビティーのデータベースしきい値によるキュー待機 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES データベースしきい値によりキューに入れられたことを示す場合に「Yes」を戻します。「No」は、アクティビティーがキューに入れられなかったことを示します。

表 249. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、アクティビティーに適用されている CONCURRENTDBCOORDACTIVITIES データベースしきい値によってそのアクティビティーがキューに入れられたかどうかを判別します。

concurrentdbcoordactivities_db_threshold_value 並行データベース・コーディネーター・アクティビティーのデータベースしきい値 : モニター・エレメント

このモニター・エレメントは、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES データベースしきい値の上限を戻します。

表 250. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES データベースしきい値がアクティビティーに適用されている場合、その値を判別します。

concurrentdbcoordactivities_db_threshold_violated 並行データベース・コーディネーター・アクティビティーのデータベースしきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES データベースしきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 251. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES データベースしきい値にアクティビティーが違反したかどうかを判別します。

concurrentdbcoordactivities_subclass_threshold_id 並行データベース・コーディネーター・アクティビティーのサービス・サブクラスしきい値 ID : モニター・エレメント

このモニター・エレメントは、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値の ID を戻します。

表 252. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値がアクティビティーに適用されていた場合、どのサービス・サブクラスしきい値が適用されていたかを判別します。

concurrentdbcoordactivities_subclass_ threshold_queued 並行データベース・コーディネーター・アクティビティーのサービス・サブクラスしきい値によるキュー待機 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値によりキューに入れられたことを示す場合に「Yes」を戻します。「No」は、アクティビティーがキューに入れられなかったことを示します。

表 253. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、アクティビティーに適用されている CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値によってそのアクティビティーがキューに入れられたかどうかを判別します。

concurrentdbcoordactivities_subclass_ threshold_value 並行データベース・コーディネーター・アクティビティーのサービス・サブクラスしきい値 : モニター・エレメント

このモニター・エレメントは、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値の上限を戻します。

表 254. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値がアクティビティーに適用されている場合、その値を判別します。

concurrentdbcoordactivities_subclass_ threshold_violated 並行データベース・コーディネーター・アクティビティーのサービス・サブクラスしきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 255. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES サービス・サブクラスしきい値にアクティビティーが違反したかどうかを判別します。

concurrentdbcoordactivities_superclass_ threshold_id 並行データベース・コーディネーター・アクティビティーのサービス・スーパークラスしきい値 ID : モニター・エレメント

アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES_SUPERCLASS しきい値の ID。

表 256. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES サービス・スーパークラスしきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

concurrentdbcoordactivities_superclass_ threshold_queued 並行データベース・コーディネーター・アクティビティーのサービス・スーパークラスしきい値によるキュー待機 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES サービス・スーパークラスしきい値によりキューに入れられたことを示す場合に「Yes」を戻します。「No」は、アクティビティーがキューに入れられなかったことを示します。

表 257. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、アクティビティーに適用されている CONCURRENTDBCOORDACTIVITIES サービス・スーパークラスしきい値によってそのアクティビティーがキューに入れられたかどうかを判別します。

concurrentdbcoordactivities_superclass_ threshold_value 並行データベース・コーディネーター・アクティビティーのサービス・スーパークラスしきい値 : モニター・エレメント

アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES サービス・スーパークラスしきい値の上限。

表 258. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES サービス・スーパークラスしきい値がアクティビティーに適用されている場合、その値を判別します。

concurrentdbcoordactivities_superclass_ threshold_violated 並行データベース・コーディネーター・アクティビティーのサービス・スーパークラスしきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES サービス・スーパークラスしきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 259. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES サービス・スーパークラスしきい値にアクティビティーが違反したかどうかを判別します。

concurrentdbcoordactivities_work_action_set_ threshold_id 並行データベース・コーディネーター・アクティビティーの作業アクション・セットしきい値 ID : モニター・エレメント

アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES 作業アクション・セットしきい値の ID。

表 260. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES 作業アクション・セットしきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

concurrentdbcoordactivities_work_action_set_ threshold_queued 並行データベース・コーディネーター・アクティビティーの作業アクション・セットしきい値によるキュー待機 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET しきい値によりキューに入れられたことを示す場合に「Yes」を戻します。「No」は、アクティビティーがキューに入れられなかったことを示します。

表 261. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、アクティビティーに適用されている CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET しきい値によってそのアクティビティーがキューに入れられたかどうかを判別します。

concurrentdbcoordactivities_work_action_set_ threshold_value 並行データベース・コーディネーター・アクティビティーの作業アクション・セットしきい値 : モニター・エレメント

アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET しきい値の上限。

表 262. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、CONCURRENTDBCOORDACTIVITIES_WORK しきい値がアクティビティーに適用されている場合、その値を判別します。

concurrentdbcoordactivities_work_action_set_ threshold_violated 並行データベース・コーディネーター・アクティビティーの作業アクション・セットしきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET しきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 263. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、アクティビティーに適用されていた CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET しきい値にアクティビティーが違反したかどうかを判別します。

conn_complete_time 接続要求完了タイム・スタンプ

接続要求が認可された日時。

エレメント ID

conn_complete_time

エレメント・タイプ

timestamp

表 264. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	タイム・スタンプ

使用法 このエレメントを使用すると、データベースへの接続要求が認可された日時を判別できます。

conn_time データベース接続時刻 : モニター・エレメント

データベースへの接続の日時 (データベース・レベルでは、これはデータベースへの最初の接続)、またはデータベースの活動化が発行された日時。

表 265. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	-
データベース	event_dbheader	-
接続	event_connheader	-

使用法

このエレメントと **disconn_time** モニター・エレメントを組み合わせて使用すると、次の時点以降の経過時間を計算できます。

- データベースがアクティブだったとき (データベース・レベルの情報の場合)。
- 接続がアクティブだったとき (接続レベルの情報の場合)。

connection_status 接続状況

このエレメントは、 GET SNAPSHOT コマンドを発行するノードと *db2nodes.cfg* ファイルにリストされているその他のノードの間の通信接続状況を示します。

エレメント ID

connection_status

エレメント・タイプ

情報

表 266. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	fcm_node	基本

使用法 次の接続値があります。

SQLM_FCM_CONNECT_INACTIVE
現在、接続されていません。

SQLM_FCM_CONNECT_ACTIVE
接続はアクティブです。

SQLM_FCM_CONNECT_CONGESTED
接続が混雑しています。

2 つのノードがアクティブな状態になっても、これらのノード間に通信がなければその間の通信接続は非アクティブとなります。

connections_top 同時接続の最大数

データベースを活動化した時点からの、そのデータベースへの同時接続の最大数。

エレメント ID

connections_top

エレメント・タイプ

水準点

表 267. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 268. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 このエレメントは、*maxappls* 構成パラメーターの設定値を評価するときに利用できます。

maxappls は、データベース接続数を制限するので、このエレメントの値が *maxappls* パラメーターと同じ場合は、一部のデータベース接続がリジェクトされていることを示します。

次の公式を使用すると、スナップショットを取った時点の接続数を計算できます。

```
rem_cons_in + local_cons
```

consistency_token パッケージ整合性トークン : モニター・エレメント

特定のパッケージ名および作成者について、複数のバージョンが存在する場合があります (DB2 バージョン 8 以降)。パッケージ整合性トークンを使用すると、現在実行中の SQL を含むパッケージのバージョンを識別できます。

表 269. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント

表 270. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
ステートメント	event_stmt	-

使用法

このエレメントを、パッケージおよび実行中の SQL ステートメントの識別に利用できます。

container_accessible - コンテナーのアクセス可能性 : モニター・エレメント

このエレメントは、コンテナーがアクセス可能かどうかを示します。値 1 は「はい」を意味し、値 0 は「いいえ」を意味します。

表 271. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE

表 272. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_container	基本

使用法 このエレメントを **container_id**、**container_name**、**container_type**、

container_total_pages、**container_usable_pages**、および **container_stripe_set** の各エレメントと組み合わせて使用して、コンテナーを記述できます。

container_id - コンテナー ID : モニター・エレメント

表スペース内のコンテナーを一意的に定義する整数。

表 273. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONTAINER 表関数 - コンテナー・メトリックの取得	DATA OBJECT METRICS BASE

表 274. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_container	基本

使用法 このエレメントと container_name、 container_type、 container_total_pages、 container_usable_pages、 container_stripe_set、および container_accessible の各エレメントを組み合わせて使用すると、コンテナーを記述できます。

container_name - コンテナー名 : モニター・エレメント

コンテナーの名前。

表 275. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONTAINER 表関数 - コンテナー・メトリックの取得	DATA OBJECT METRICS BASE

表 276. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_container	基本

使用法 このエレメントと container_id、 container_type、 container_total_pages、 container_usable_pages、 container_stripe_set、および container_accessible の各エレメントを組み合わせて使用すると、コンテナーを記述できます。

container_stripe_set - コンテナー・ストライプ・セット : モニター・エレメント

コンテナーが属するストライプ・セット。

表 277. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE

表 278. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_container	基本

使用法

このモニター・エレメントを **container_id**、**container_name**、**container_type**、**container_total_pages**、**container_usable_pages**、および **container_accessible** の各エレメントと組み合わせて使用して、コンテナーを記述します。これは、DMS 表スペースにのみ適用できます。

container_total_pages - コンテナー内の合計ページ数 : モニター・エレメント

コンテナーが占有するページ数の合計。

表 279. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE

表 280. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_container	基本 (DMS 表スペース) バッファー・プール (SMS 表スペース)

使用法 このエレメントと container_id、 container_name、 container_type、 container_usable_pages、 container_stripe_set、および container_accessible の各エレメントを組み合わせて使用すると、コンテナーを記述できます。

container_type - コンテナー・タイプ : モニター・エレメント

コンテナーのタイプ。

表 281. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_CONTAINER 表関数 - コンテナー・メトリックの取得	DATA OBJECT METRICS BASE

表 282. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_container	基本

使用法

このエレメントはコンテナーのタイプを戻します。コンテナーのタイプは、ディレクトリー・パス (SMS の場合のみ)、ファイル (DMS の場合)、ロー・デバイス (DMS の場合) のいずれかです。このエレメントを、**container_id**、**container_name**、**container_total_pages**、**container_usable_pages**、**container_stripe_set**、および **container_accessible** の各エレメントと組み合わせて使用して、コンテナーを記述できます。

このモニター・エレメントの有効な値は sqlutil.h ファイルに定義されています。

container_usable_pages - コンテナー内の使用可能なページ数 : モニター・エレメント

コンテナー内の使用可能なページ数の合計。

表 283. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE

表 284. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_container	基本 (DMS 表スペース) バッファー・プール (SMS 表スペース)

使用法 このエレメントと container_id、 container_name、 container_type、 container_total_pages、 container_stripe_set、および container_accessible の各エレメントを組み合わせて使用すると、コンテナーを記述できます。 SMS 表スペースの場合、この値は container_total_pages と同じになります。

coord_act_aborted_total 打ち切られたコーディネーター・アクティビティーの合計 : モニター・エレメント

最後にリセットしてからの、エラーで完了した任意のネスト・レベルのコーディネーター・アクティビティーの合計数。サービス・クラスでは、アクティビティーの完了時に値は更新されます。ワークロードでは、その作業単位の最後に各ワークロード・オカレンスによって値が更新されます。

サービス・クラスでは、アクティビティーが異常終了前に REMAP ACTIVITY アクションで別のサービス・サブクラスに再マップされた場合、このアクティビティーのカウントは、異常終了時のサブクラスでの合計にのみ含められます。

表 285. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-

表 285. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats	-

使用法

このエレメントを使用して、システム上のアクティビティーが正常に完了しているかを知ることができます。アクティビティーは、取り消し、エラー、または反作用しきい値のために打ち切られる場合があります。

coord_act_completed_total 完了したコーディネーター・アクティビティーの合計 : モニター・エレメント

最後にリセットしてからの、正常に完了した任意のネスト・レベルのコーディネーター・アクティビティーの合計数。サービス・クラスでは、アクティビティーの完了時に値は更新されます。ワークロードでは、その作業単位の最後に各ワークロード・オカレンスによって値が更新されます。

サービス・クラスでは、アクティビティーが完了前に REMAP ACTIVITY アクションで別のサービス・サブクラスに再マップされた場合、このアクティビティーのカウントは、完了時のサブクラスでの合計にのみ含められます。

表 286. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats	-
統計	event_scstats	-

使用法

このエレメントを使用すると、システムのアクティビティーのスループットを判別したり、複数のパーティション間の平均アクティビティー存続時間の計算を補助したりすることができます。

coord_act_est_cost_avg コーディネーター・アクティビティーの平均見積コスト : モニター・エレメント

最後のリセット以降に、このサービス・サブクラスまたは作業クラスに関連付けられた、ネスト・レベル 0 のコーディネーター DML アクティビティーの見積コストの算術平均。内部的に追跡されている平均がオーバーフローすると、値 -2 が返されます。サービス・サブクラスでは、そのサービス・サブクラスの COLLECT AGGREGATE ACTIVITY DATA が NONE または BASE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA EXTENDED 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE または BASE に設定されている場合、このモニター・エレメントは -1 を返します。単位はミリ秒です。

サービス・クラスの場合、アクティビティーの見積コストは、アクティビティーがシステムに入るのに使用するサービス・サブクラスでしかカウントされません。REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーを再マップしたサービス・サブクラスの coord_act_est_cost_avg 平均は影響されません。

表 287. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

この統計を使用すると、最後の統計リセット以降に完了または異常終了したこのサービス・サブクラス、ワークロード、または作業クラスに関連付けられた、ネスト・レベル 0 のコーディネーター DML アクティビティーの見積コストの算術平均を判別できます。

この平均を使用して、アクティビティー見積コストのヒストグラムに使用されるヒストグラム・テンプレートが適切かどうかを判別することもできます。アクティビティー見積コストのヒストグラムからアクティビティーの平均見積コストを計算してください。計算した平均をこのモニター・エレメントと比較してください。計算した平均が、このモニター・エレメントによって報告される真の平均から大きく外れるようなら、データにより適切なビン値のセットを使用する、アクティビティー見積コストのヒストグラムに関するヒストグラム・テンプレートに変更することを考慮してください。

coord_act_exec_time_avg コーディネーター・アクティビティー平均実行時間 : モニター・エレメント

最後のリセット以降に、このサービス・サブクラスまたは作業クラスに関連付けられた、ネスト・レベル 0 のコーディネーター・アクティビティーの実行時間の算術平均。内部的に追跡されている平均がオーバーフローすると、値 -2 が返されます。サービス・サブクラスでは、そのサービス・サブクラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。 単位はミリ秒です。

サービス・クラスの場合、REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーがマップされたが完了しなかったサービス・サブクラスの coord_act_exec_time_avg 平均は影響されません。

表 288. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

この統計を使用すると、完了または異常終了したサービス・サブクラス、ワークロード、または作業クラスに関連付けられたコーディネーター・アクティビティーの実行時間の算術平均を判別できます。

この平均を使用して、アクティビティー実行時間のヒストグラムに使用されるヒストグラム・テンプレートが適切かどうかを判別することもできます。アクティビティー実行時間のヒストグラムから平均アクティビティー実行時間を計算してください。計算した平均をこのモニター・エレメントと比較してください。計算した平均が、このモニター・エレメントによって報告される真の平均から大きく外れるようなら、データにより適切なビン値のセットを使用する、アクティビティー実行時間のヒストグラムに関するヒストグラム・テンプレートに変更することを考慮してください。

coord_act_interarrival_time_avg コーディネーター・アクティビティーの平均到着時間 : モニター・エレメント

最後のリセット以降に、このサービス・サブクラスまたは作業クラスに関連付けられた、ネスト・レベル 0 のコーディネーター・アクティビティーの到着間隔の時間の算術平均。内部的に追跡されている平均がオーバーフローすると、値 -2 が返されます。サービス・サブクラスでは、そのサービス・サブクラスの COLLECT AGGREGATE ACTIVITY DATA が NONE または BASE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA EXTENDED 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE または BASE に設定されている場合、このモニター・エレメントは -1 を返します。 単位はミリ秒です。

サービス・クラスの場合、inter-arrival 時間の平均は、アクティビティーがシステムに入るのに使用するサービス・サブクラスから計算されます。 REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーを再マップしたサービス・サブクラスの coord_act_interarrival_time_avg は影響されません。

表 289. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

この統計を使用すると、このサービス・サブクラス、ワークロード、または作業クラスに関連付けられた、ネスト・レベル 0 のコーディネーター・アクティビティーの到着間隔の算術平均を判別できます。

到着間隔の時間を使用して、到着レートを判別できます。到着レートは到着間隔の時間の逆になります。この平均を使用して、アクティビティー到着間隔の時間のヒストグラムに使用されるヒストグラム・テンプレートが適切かどうかを判別することもできます。アクティビティー到着間隔の時間のヒストグラムから平均アクティビティー到着間隔の時間を計算してください。計算した平均をこのモニター・エレメントと比較してください。計算した平均が、このモニター・エレメントによって報告される真の平均から大きく外れるようなら、データにより適切なビン値のセットを使用する、アクティビティー到着間隔の時間のヒストグラムに関するヒストグラム・テンプレートに変更することを考慮してください。

coord_act_lifetime_avg コーディネーター・アクティビティー存続時間の平均 : モニター・エレメント

最後のリセット以降に、このサービス・サブクラス、ワークロード、または作業クラスに関連付けられた、ネスト・レベル 0 のコーディネーター・アクティビティーの存続時間の算術平均。内部的に追跡されている平均がオーバーフローすると、値 -2 が返されます。サービス・サブクラスでは、そのサービス・サブクラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。 単位はミリ秒です。

サービス・クラスの場合、REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーが完了した最後のサービス・クラスの coord_act_lifetime_avg 平均のみが影響されます。

表 290. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

この統計を使用すると、完了または異常終了したサービス・サブクラス、ワークロード、または作業クラスに関連付けられたコーディネーター・アクティビティーの存続時間の算術平均を判別できます。

この統計を使用して、アクティビティー存続時間のヒストグラムに使用されるヒストグラム・テンプレートが適切かどうかを判別することもできます。アクティビティー存続時間のヒストグラムから平均アクティビティー存続時間を計算してくださ

い。計算した平均をこのモニター・エレメントと比較してください。計算した平均が、このモニター・エレメントによって報告される真の平均から大きく外れるようなら、データにより適切なビン値のセットを使用する、アクティビティー存続時間のヒストグラムに関するヒストグラム・テンプレートに変更することを考慮してください。

coord_act_lifetime_top コーディネーター・アクティビティー存続時間の最上位 : モニター・エレメント

すべてのネスト・レベルでカウントされる、コーディネーター・アクティビティー存続時間の最高水準点。単位はミリ秒です。サービス・クラスでは、サービス・クラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。

REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップする際に、サービス・クラスでこの統計を効果的に使用するには、与えられたサービス・サブクラスの coord_act_lifetime_top 最高水準点と、同じ再マップのしきい値 (複数可) によって影響される他のサブクラスの最高水準点を集約する必要があります。 これは、アクティビティーが完了するのは、再マップしきい値によって別のサービス・サブクラスへ再マップされた後になるからであり、アクティビティーが再マップされるまで他のサービス・サブクラスにいた時間は、完了時のサブクラスでの合計にのみ含められるからです。

表 291. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wcstats	-
統計	event_scstats	-
統計	event_wlstats	-

使用法

このエレメントを使用すると、アクティビティー存続時間のしきい値が有効であるかどうかを判別する助けになります。さらに、そのようなしきい値を構成する方法を判別する助けとすることもできます。

coord_member - コーディネーター・メンバー : モニター・エレメント

指定された作業単位またはワークロードのコーディネーター・メンバー。

表 292. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	常に収集される

表 292. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	常に収集される
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

coord_act_queue_time_avg コーディネーター・アクティビティー・キュー平均時間 : モニター・エレメント

最後のリセット以降に、このサービス・サブクラスまたは作業クラスに関連付けられた、ネスト・レベル 0 のコーディネーター・アクティビティーのキュー時間の算術平均。内部的に追跡されている平均がオーバーフローすると、値 -2 が返されます。サービス・サブクラスでは、そのサービス・サブクラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。 単位はミリ秒です。

サービス・クラスの場合、キュー時間はアクティビティーが完了するもしくは異常終了したサービス・サブクラスでしかカウントされません。 REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーがマップされたが完了しなかったサービス・サブクラスの coord_act_queue_time_avg 平均は影響されません。

エレメント ID

coord_act_queue_time_avg

エレメント・タイプ

情報

表 293. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

この統計を使用すると、完了または異常終了したサービス・サブクラス、ワークロード、または作業クラスに関連付けられたコーディネーター・アクティビティーのキュー時間の算術平均を判別できます。

この統計を使用して、アクティビティー・キュー時間のヒストグラムに使用されるヒストグラム・テンプレートが適切かどうかを判別することもできます。アクティ

ビティー・キュー時間のヒストグラムから平均アクティビティー・キュー時間を計算してください。計算した平均をこのモニター・エレメントと比較してください。計算した平均が、このモニター・エレメントによって報告される真の平均から大きく外れるようなら、データにより適切なビン値のセットを使用する、アクティビティー・キュー時間のヒストグラムに関するヒストグラム・テンプレートに変更することを考慮してください。

coord_act_rejected_total リジェクトされたコーディネーター・アクティビティーの合計 : モニター・エレメント

最後にリセットしてからの、実行が許可されず、リジェクトされた任意のネスト・レベルのコーディネーター・アクティビティーの合計数。このカウンターは、予測しきい値または実行阻止作業アクションのいずれかによりアクティビティーの実行が阻止された場合に更新されます。サービス・クラスでは、アクティビティーの完了時に値は更新されます。ワークロードでは、その作業単位の最後に各ワークロード・オカレンスによって値が更新されます。

表 294. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wlstats	-

使用法

このエレメントを使用すると、予測しきい値および実行を阻止する作業アクションが有効であるかどうか、および、それらの制限が大きすぎないかどうかを判別する助けになります。

coord_agent_pid コーディネーター・エージェント ID : モニター・エレメント

アプリケーションに関するコーディネーター・エージェントのエンジン・ディスパッチ可能単位 (EDU) ID。Linux オペレーティング・システムを除いて、EDU ID はスレッド ID にマップされます。Linux オペレーティング・システムでは、EDU ID は DB2 生成のユニーク ID です。

表 295. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本

表 296. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-

使用法

このエレメントを使用すると、データベース・システム・モニター情報をシステム・トレースなどの他の診断情報のソースにリンクできます。

coord_agents_top - コーディネーター・エージェント最大数 :

同時に動作できるコーディネーター・エージェントの最大数。

表 297. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本
データベース	dbase	基本

使用法

コーディネーター・エージェントの最大値がこのノードのワークロードとして大きすぎる場合は、**max_coordagents** 構成パラメーターを変更することで、この上限を下げることができます。

coord_node コーディネーター・ノード

マルチノード・システムでは、インスタンスに接続つまりアタッチされたアプリケーションがあるノードのノード番号。

表 298. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

表 299. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	-

使用法 接続されたアプリケーションは、それぞれ 1 つのコーディネーター・ノードにより処理されます。

coord_partition_num コーディネーター・パーティション番号 : モニター・エレメント

作業単位またはアクティビティーのコーディネーター・パーティション。複数パーティションのシステムでは、コーディネーター・パーティションは、アプリケーションがデータベースに接続したパーティションです。

表 300. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	-
アクティビティー	event_activity	-

表 300. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-

使用法

このエレメントを使用して、コーディネーター以外のパーティションにレコードがあるアクティビティーまたは作業単位の、コーディネーター・パーティションを識別できます。

corr_token DRDA 相関トークン

DRDA AS 相関トークン。

表 301. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
アプリケーション	appl	基本

表 302. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_connheader	-

使用法 DRDA 相関トークンは、アプリケーション・サーバーとアプリケーション・リクエスターの間の処理の相関を求めるときに使用します。これはエラーが発生したときにログにダンプされる ID であるため、エラーとなった会話を識別するのに利用できます。場合によっては、会話の LUWID を示します。

通信に DRDA を使用していない場合は、このエレメントが *appl_id* を戻します (appl_id 参照)。

データベース・システム・モニター API を使用すると、このエレメントの長さを判別するために API 定数の SQLM_APPLID_SZ が使用されることに注意してください。

cost_estimate_top コスト見積もりの最上位 : モニター・エレメント

サービス・サブクラスまたは作業クラスでの、すべてのネスト・レベルにおける DML アクティビティーの見積コストの最高水準点。サービス・サブクラスでは、そのサービス・サブクラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。

サービス・クラスの場合、DML アクティビティーの見積コストは、アクティビティーがシステムに入るのに使用するサービス・サブクラスでしかカウントされません。 REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティ

ビティーを再マップすると、アクティビティーを再マップしたサービス・サブクラスの cost_estimate_top は影響されません。

表 303. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

このエレメントを使用して、収集された時間間隔にサービス・クラス、ワークロード、または作業クラス用のパーティションで到達した DML アクティビティー見積コストの最大値を判別することができます。

count イベント・モニター・オーバーフロー数

連続して発生したオーバーフローの数。

エレメント ID
count

エレメント・タイプ
カウンター

表 304. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
オーバーフロー・レコード	event_overflow	-

使用法 このエレメントを使用すると、失われたモニター・データの量がわかります。

イベント・モニターは、一連のオーバーフローについて 1 つのオーバーフロー・レコードを送信します。

cputime_threshold_id - CPU 時間しきい値 ID : モニター・エレメント

アクティビティーに適用されていた CPUTIME しきい値の ID。

表 305. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、 CPUTIME しきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

cputime_threshold_value - CPU 時間しきい値 : モニター・エレメント

アクティビティーに適用されていた CPUTIME しきい値の上限。

表 306. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、CPUTIME しきい値がアクティビティーに適用されている場合、その値を判別します。

cputime_threshold_violated - CPU 時間しきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CPUTIME しきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 307. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、アクティビティーに適用されていた CPUTIME しきい値にアクティビティーが違反したかどうかを判別します。

cputimeinsc_threshold_id - サービス・クラス内の CPU 時間しきい値 ID : モニター・エレメント

アクティビティーに適用されていた CPUTIMEINSC しきい値の ID。

表 308. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、 CPUTIMEINSC しきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

cputimeinsc_threshold_value - サービス・クラス内の CPU 時間しきい値 : モニター・エレメント

アクティビティーに適用されていた CPUTIMEINSC しきい値の上限。

表 309. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、CPUTIMEINSC しきい値がアクティビティーに適用されている場合、その値を判別します。

cputimeinsc_threshold_violated - サービス・クラス内の CPU 時間しきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが CPUTIMEINSC しきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 310. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、アクティビティーに適用されていた CPUTIMEINSC しきい値にアクティビティーが違反したかどうかを判別します。

create_nickname ニックネーム作成回数

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時か、またはデータベース・モニター・カウンターの最後のリセット時のどちらか最近の方の時点以降に、いずれかのアプリケーションに代わってフェデレーテッド・サーバーがこのデータ・ソース上にあるオブジェクトのニックネームを作成した合計回数が含まれています。

表 311. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

このエレメントを使用すると、このフェデレーテッド・サーバー・インスタンスまたはアプリケーションによるこのデータ・ソースへの CREATE NICKNAME アクティビティーの量を判別できます。 CREATE NICKNAME 処理を行うと、データ・ソース・カタログに対して複数の照会が実行されるので、このエレメントの値が大きい場合は、原因を突き止めるか、またはアクティビティーを制限する必要があります。

create_nickname_time ニックネーム作成応答時間

このエレメントには、このフェデレーテッド・サーバー・インスタンス上で実行されているすべてのアプリケーションまたは単一アプリケーションからの CREATE NICKNAME ステートメントを、このデータ・ソースが処理するのに要した合計時間が含まれています (ミリ秒単位)。この応答時間は、フェデレーテッド・サーバー・インスタンスの開始時か、またはデータベース・モニター・カウンターの最後のリセット時のどちらか最近の方の時点以降から測定されます。応答時間は、フェデレーテッド・サーバーが CREATE NICKNAME ステートメントを処理するためにデータ・ソースから情報の検索を開始してから必要なデータの取り出しがすべて完了するまでの時間です。

表 312. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	タイム・スタンプ
アプリケーション	appl_remote	タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、このデータ・ソースでニックネームを作成するのに要した実際の時間を判別できます。

creator アプリケーション作成者

アプリケーションをプリコンパイルしたユーザーの許可 ID。

表 313. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント
DCS ステートメント	dcs_stmt	ステートメント

表 314. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	event_dlconn	-
ステートメント	event_stmt	-
アクティビティー	event_activitystmt	-

使用法 このエレメントとカタログ内のパッケージ・セクション情報の CREATOR 列を組み合わせて使用し、処理中の SQL ステートメントを識別してください。

CURRENT PACKAGE PATH 特殊レジスターを設定した場合には、 SQL ステートメントの存続期間中に *creator* 値はさまざまな値を反映します。 PACKAGE PATH の解決の前にスナップショットまたはイベント・モニター・レコードが取られる場合は、 *creator* 値はクライアント要求から流れる値を反映します。 PACKAGE PATH の解決の後にスナップショットまたはイベント・モニター・レコードが取られる場合は、 *creator* 値は解決されたパッケージの作成者を反映します。解決されたパッケージの *creator* 値は CURRENT PACKAGE PATH SPECIAL REGISTER 中に最初に表示される値で、パッケージ名およびユニーク ID はクライアント要求の名前および ID と一致します。

current_active_log 現行アクティブ・ログ・ファイル番号

現在 DB2 データベース・システムが書き込んでいるアクティブ・ログ・ファイルのファイル番号。

表 315. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	detail_log	基本

表 316. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 このエレメントと *first_active_log* および *last_active_log* エレメントを組み合わせて使用すると、アクティブ・ログ・ファイルの範囲を判別できます。アクティブ・ログ・ファイルの範囲を知っていると、ログ・ファイルに必要なディスク・スペースを判別するのに役立ちます。

また、このエレメントを使用すると、どのログ・ファイルにデータがあるか判別でき、スプリット・ミラー・サポートが必要なログ・ファイルを識別するのに役立ちます。

current_archive_log 現行アーカイブ・ログ・ファイル番号

現在 DB2 データベース・システムがアーカイブしているログ・ファイルのファイル番号。 DB2 データベース・システムがログ・ファイルをアーカイブしていない場合は、このエレメントの値は SQLM_LOGFILE_NUM_UNKNOWN になります。

表 317. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	detail_log	基本

表 318. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 このエレメントを使用して、ログ・ファイルのアーカイブに問題があるかどうかを判別します。この種の問題には、以下のものがあります。

- 低速のアーカイブ・メディア
- 使用不可であるアーカイブ・メディア

current_extent - 現在移動中のエクステント : モニター・エレメント

表スペースのリバランス処理によって現在移動中であるエクステントの数値 ID。

表 319. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_EXTENT_MOVEMENT_STATUS - エクステントの移動の進行状況メトリックの取得	常に収集される

cursor_name カーソル名

この SQL ステートメントに対応するカーソルの名前。

エレメント ID

cursor_name

エレメント・タイプ

情報

表 320. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント

表 321. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
詳細付きデッドロック	event_detailed_dlconn	-
ステートメント	event_stmt	-

使用法 このエレメントを使用すると、処理中の SQL ステートメントを識別できます。この名前は、SQL SELECT ステートメントの OPEN、 FETCH、CLOSE、および PREPARE に使用されます。カーソルを使用しない場合は、このフィールドはブランクになります。

data_object_pages データ・オブジェクト・ページ数

表が使用するディスク・ページの数。このサイズは、基本表のサイズのみを表します。索引オブジェクト、LOB データ、および LONG データが使用するスペースは、それぞれ *index_object_pages*、 *lob_object_pages*、および *long_object_pages* によって報告されます。

表 322. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本

表 323. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表	event_table	-

使用法 このエレメントを使用すると、特定の表が使用する実際のスペースの量を表示できます。このエレメントと表イベント・モニターを組み合わせて使用すると、時間とともに表が大きくなる比率を追跡できます。

data_partition_id - データ・パーティション ID : モニター・エレメント

情報が戻されるデータ・パーティションの ID。

表 324. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	常に収集される
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

表 325. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本
ロック	lock	ロック
ロック	lock_wait	ロック

表 326. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表	event_table	-
デッドロック	event_dlconn	-
詳細付きデッドロック	event_detailed_dlconn	-
デッドロック	lock	-

使用法

このエレメントは、パーティション表およびパーティション化索引にのみ適用されます。その他の場合、このモニター・エレメントの値は NULL です。

ロック・レベル情報が戻されるとき、-1 という値は、表全体へのアクセスを制御するロックを表します。

datasource_name データ・ソース名

このエレメントには、フェデレーテッド・サーバーによってリモート・アクセス情報が表示されているデータ・ソースの名前が含まれています。このエレメントは、SYSCAT.SERVERS の 'SERVER' 列に対応しています。

エレメント ID

datasource_name

エレメント・タイプ

情報

表 327. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	基本
アプリケーション	appl_remote	基本

使用法 このエレメントを使用すると、アクセス情報が収集されて戻されているデータ・ソースを識別できます。

db2_status DB2 インスタンス状況

データベース・マネージャーのインスタンスの現在の状況。

表 328. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法 このエレメントを使用して、データベース・マネージャー・インスタンスの状態を判別できます。

このエレメントの値は以下のとおりです。

API 定数	値	説明
SQLM_DB2_ACTIVE	0	データベース・マネージャー・インスタンスはアクティブになっている。

API 定数	値	説明
SQLM_DB2_QUIESCE_PEND	1	インスタンス内のインスタンスおよびデータベースは静止ペンディング状態となっている。インスタンス・データベースへの新規接続は許可されず、新規作業単位を開始することはできません。静止要求によっては、アクティブな作業単位の完了が許可される場合と即時ロールバックが行われる場合があります。
SQLM_DB2_QUIESCED	2	インスタンス内のインスタンスおよびデータベースは静止状態となっている。インスタンス・データベースへの新規接続は許可されず、新規作業単位を開始することはできません。

db2start_time データベース・マネージャー開始タイム・スタンプ

db2start コマンドを使用してデータベース・マネージャーを開始した日時。

エレメント ID

db2start_time

エレメント・タイプ

timestamp

表 329. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法 このエレメントと *time_stamp* モニター・エレメントを組み合わせて使用すると、データベース・マネージャーを開始してからスナップショットが取られるまでの経過時間を計算できます。

db_conn_time データベース活動化タイム・スタンプ : モニター・エレメント

データベースへの接続の日時 (データベース・レベルでは、これはデータベースへの最初の接続)、またはデータベースの活動化が発行された日時。

表 330. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	タイム・スタンプ
表スペース	tablespace_list	バッファー・プール、タイム・スタンプ
表	table_list	タイム・スタンプ

表 331. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	-

使用法

このエレメントと **disconn_time** モニター・エレメントを組み合わせて使用すると、合計接続時間を計算できます。

db_heap_top 割り振られた最大データベース・ヒープ

このエレメントは、 DB2 のバージョン間での互換性を確保するために維持されています。現在は、メモリーの使用量を計算しますが、データベース・ヒープの使用量だけが対象ではありません。

注: db_heap_top モニター・エレメントは、DB2 バージョン 9.5 以降では非推奨になっています。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 332. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 333. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

db_location データベース・ロケーション

アプリケーションに関連したデータベースのロケーション。

表 334. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 スナップショットをとるアプリケーションに対応する、データベース・サーバーの相対的なロケーションを判別します。次の値があります。

- SQLM_LOCAL
- SQLM_REMOTE

db_name データベース名

情報が収集されるデータベースの実名、またはアプリケーションの接続先のデータベースの実名。これはデータベースの作成時に与えられた名前です。

表 335. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
データベース	dbase_remote	基本
アプリケーション	appl_id_info	基本
アプリケーション	appl_remote	基本
表スペース	tablespace_list	バッファー・プール
バッファー・プール	バッファー・プール	バッファー・プール
表	table_list	表
ロック	db_lock_list	基本
動的 SQL	dynsql_list	基本
DCS データベース	dcs_dbase	基本
DCS アプリケーション	dcs_appl_info	基本

表 336. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_dbheader	-

使用法 このエレメントを使用すると、データが適用される特定のデータベースを識別できます。

ホストへの接続、または System i® のデータベース・サーバーへの接続で DB2 Connect を使用しないアプリケーションの場合は、このエレメントと **db_path** モニター・エレメントを組み合わせて使用すると、データベースを個別に識別し、モニターが提供する情報の各レベルに関連付けることができます。

db_path データベース・パス

モニター対象のシステムに保管されているデータベースのロケーションを示す絶対パスです。

表 337. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl_id_info	基本
表スペース	tablespace_list	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
表	table_list	表
ロック	db_lock_list	基本
動的 SQL	dynsql_list	基本

表 338. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_dbheader	-

使用法 このエレメントと *db_name* モニター・エレメントを組み合わせて使用すると、データが適用される特定のデータベースを識別できます。

db_status データベース状況

データベースの現在の状況。

表 339. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 このエレメントを使用して、データベースの状態を判別できます。

このフィールドの値は以下のとおりです。

API 定数	値	説明
SQLM_DB_ACTIVE	0	データベースはアクティブになっている。
SQLM_DB_QUIESCE_PEND	1	データベースは静止ペンディング状態となっている。データベースに対する新しい接続は許可されません。新しい作業単位も開始できません。静止要求によっては、アクティブな作業単位の完了が許可される場合と即時ロールバックが行われる場合があります。
SQLM_DB_QUIESCED	2	データベースは静止状態となっている。データベースに対する新しい接続は許可**されません**。新しい作業単位も開始できません。
SQLM_DB_ROLLFWD	3	データベースでロールフォワードが進行中。

db_storage_path 自動ストレージ・パス : モニター・エレメント

このエレメントは、自動ストレージ表スペースを配置するためにデータベースによって使用されるロケーションの絶対パスを示します。データベースに関連したストレージ・パスは 0 個以上あります。

表 340. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	db_sto_path_info	基本

使用法

このエレメントを **num_db_storage_paths** モニター・エレメントと一緒に使用して、このデータベースに関連したストレージ・パスを識別します。

db_storage_path_state - ストレージ・パスの状態 : モニター・エレメント

自動ストレージ・パスの状態は、ストレージ・パスがデータベースによって使用されているかどうかを示します。

表 341. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	db_sto_path_info	基本

使用法

このモニター・エレメントを使用して、ストレージ・パスがデータベースによって使用されているかどうか判別します。次の値が使用されます。

NOT_IN_USE
指定されたデータベース・パーティションに、このストレージ・パスを使用している表スペースはありません。

IN_USE
指定されたデータベース・パーティションに、このストレージ・パスを使用している表スペースがあります。

DROP_PENDING
このストレージ・パスはドロップされましたが、まだ使用している表スペースがあります。ストレージ・パスがデータベースから物理的にドロップされる前に、すべての表スペースはその使用を停止しなければなりません。ドロップされたストレージ・パスの使用を停止するには、表スペースをドロップするか、または ALTER TABLESPACE ステートメントの REBALANCE 節を使用して表スペースのリバランスを行います。

db_storage_path_with_dpe - データベース・パーティション式を含むストレージ・パス : モニター・エレメント

未評価のデータベース・パーティション式を含む自動ストレージ・パス。

表 342. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	db_sto_path_info	基本

使用法

ストレージ・パスにデータベース・パーティション式が含まれる場合、このモニター・エレメントを使用して、CREATE DATABASE コマンドまたは ALTER DATABASE ステートメントの一部としてデータベースに対して指定されたストレージ・パスを判別します。

ストレージ・パスにデータベース・パーティション式が含まれない場合は、このモニター・エレメントは NULL 値を戻します。

db_work_action_set_id データベース作業アクション・セット ID : モニター・エレメント

このアクティビティーがデータベース有効範囲の作業クラスにカテゴリー化されている場合、このモニター・エレメントは、この作業クラスが所属する作業クラス・セットに関連した作業アクション・セットの ID を示します。それ以外の場合、このモニター・エレメントは 0 の値を示します。

表 343. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
WLM_GET_ACTIVITY_DETAILS_COMPLETE (DETAILS XML 文書に報告されます)	常に収集される

表 344. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

使用法

このエレメントと **db_work_class_id** エレメントを組み合わせて使用すると、アクティビティーのデータベース作業クラスが存在する場合にはそれを一意的に識別できます。

db_work_class_id データベース作業クラス ID : モニター・エレメント

このアクティビティーがデータベース有効範囲の作業クラスにカテゴリー化されている場合、このモニター・エレメントは、この作業クラスの ID を表示します。それ以外の場合、このモニター・エレメントは 0 の値を表示します。

表 345. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
WLM_GET_ACTIVITY_DETAILS_COMPLETE 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

表 346. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

使用法

このエレメントと **db_work_action_set_id** エレメントを組み合わせて使用すると、アクティビティーのデータベース作業クラスが存在する場合にはそれを一意的に識別できます。

dcs_appl_status DCS アプリケーション状況

DB2 Connect ゲートウェイでの DCS アプリケーションの状況。

表 347. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl_info	基本

使用法 このエレメントは、DCS アプリケーションに関する問題判別に使用します。次の値があります。

- SQLM_DCS_CONNECTPEND_OUTBOUND

  アプリケーションが DB2 Connect ゲートウェイからホスト・データベースへのデータベース接続を開始しましたが、要求はまだ完了していません。

- SQLM_DCS_UOWWAIT_OUTBOUND

  DB2 Connect ゲートウェイは、ホスト・データベースがアプリケーションの要求に応答するのを待っています。

- SQLM_DCS_UOWWAIT_INBOUND

  DB2 Connect ゲートウェイからホスト・データベースへの接続が確立され、ゲートウェイがアプリケーションの SQL 要求を待っています。あるいは、アプリケーション内の作業単位に代わって、 DB2 Connect ゲートウェイが待機しています。通常、これはアプリケーションのコードが実行中であることを意味します。

dcs_db_name DCS データベース名

DCS ディレクトリーにカタログされている DCS データベースの名前。

表 348. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	基本
DCS アプリケーション	dcs_appl_info	基本

使用法 このエレメントは、DCS アプリケーションに関する問題判別に使用します。

ddl_sql_stmts データ定義言語 (DDL) SQL ステートメント

このエレメントは、実行された SQL データ定義言語 (DDL) ステートメントの数を示します。

表 349. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 350. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 このエレメントを使用すると、アプリケーション・レベルまたはデータベース・レベルのデータベース・アクティビティーのレベルを判別できます。DDL ステートメントは、システム・カタログ表への影響のために実行にコストがかかります。そのため、このエレメントの値が大きい場合は、その原因を突き止めて、このアクティビティーが実行されないように制約すべきです。

このエレメントを使用すると、次の公式を使用して、 DDL アクティビティーのパーセンテージも計算できます。

```
ddl_sql_stmts / total number of statements
```

この情報は、アプリケーションのアクティビティーおよびスループットの分析に役立ちます。 DDL ステートメントも次の項目に影響を与えます。

- カタログ・キャッシュ。保管されている表記述子情報と許可情報が無効になるので、システム・カタログから情報を取り出すためのシステム・オーバーヘッドが増加します。
- パッケージ・キャッシュ。保管されているセクションが無効になるので、セクションの再コンパイルのためのシステム・オーバーヘッドが増加します。

DDL ステートメントの例としては、CREATE TABLE、CREATE VIEW、ALTER TABLE、および DROP INDEX があります。

deadlock_id デッドロック・イベント ID

デッドロックのデッドロック ID。

エレメント ID

deadlock_id

エレメント・タイプ

情報

表 351. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	event_deadlock	-
デッドロック	event_dlconn	-
詳細付きデッドロック	event_detailed_dlconn	-
詳細付きデッドロック履歴	event_detailed_dlconn	-
詳細付きデッドロック履歴	event_stmt_history	-
詳細付きデッドロック履歴値	event_data_value	-
詳細付きデッドロック履歴値	event_detailed_dlconn	-

表 351. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
詳細付きデッドロック履歴値	event_stmt_history	-

使用法 モニター・アプリケーションでこのエレメントを使用すると、デッドロック接続およびステートメント履歴イベント・レコードと、デッドロック・イベント・レコードとを関連付けることができます。

deadlock_node デッドロック発生場所のパーティション番号

デッドロックが発生した場所のパーティション番号。

エレメント ID

deadlock_node

エレメント・タイプ

情報

表 352. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	event_deadlock	-
デッドロック	event_dlconn	-
詳細付きデッドロック	event_detailed_dlconn	-

使用法 このエレメントが該当するのは、パーティション・データベースだけです。モニター・アプリケーションでこのエレメントを使用すると、デッドロック接続イベント・レコードとデッドロック・イベント・レコードを関連付けることができます。

deadlocks - デッドロック検出数 : モニター・エレメント

発生したデッドロックの合計数。

表 353. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 353. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 354. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	ロック

スナップショット・モニターの場合、このカウンターはリセットできます。

表 355. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
接続	event_conn	-

使用法

このエレメントは、アプリケーション間で競合の問題が起きていることを示す場合があります。問題の原因としては、次の状態が考えられます。

- データベースでロック・エスカレーションが発生している場合。
- システムが生成した行のロッキング数が十分なときに、アプリケーションが表を明示的にロッキングしている場合。
- アプリケーションがバインディングのときに不適切な分離レベルを使用している場合。

- カタログ表が反復可能読み取りのためにロックされている場合。
- 複数のアプリケーションが同じロックを異なる順序で獲得しているために、デッドロックになっている場合。

この問題は、デッドロックが発生しているアプリケーション (またはアプリケーション処理) が判別できれば解決できます。この場合、アプリケーションが並行して実行できるようにアプリケーションを変更できます。ただし、一部のアプリケーションでは並行して実行できない場合があります。

接続タイム・スタンプ・モニター・エレメント (**last_reset**、**db_conn_time**、および **appl_con_time**) を使用すると、デッドロックの重大度を判別できます。例えば、デッドロックが 5 時間に 10 回起こるよりも、5 分間に 10 回起こるほうが重大です。

上記の関連エレメントについての記述部分では、調整に関するその他の推奨事項が示されています。

degree_parallelism 並列処理の度合い

照会がバインドされたときに要求された並列処理の度合い。

エレメント ID
degree_parallelism

エレメント・タイプ
情報

表 356. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント

使用法 agents_top と組み合わせて使用すると、照会で最大レベルの並列処理ができたかどうかを判別できます。

del_keys_cleaned - クリーンアップされた疑似削除キー : モニター・エレメント

クリーンアップされた疑似削除キーの数。

表 357. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

delete_sql_stmts 削除回数

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時か、またはデータベース・モニター・カウンターの最後のリセット時のどちらか最近の方の時点以降に、いずれかのアプリケーションに代わってフェデレーテッド・サーバーがこのデータ・ソースに DELETE ステートメントを発行した合計回数が含まれています。

表 358. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、フェデレーテッド・サーバーまたはアプリケーションによりこのデータ・ソースに対して行われたデータベース・アクティビティーのレベルを判別できます。

このエレメントを使用すると、次の公式を使用して、フェデレーテッド・サーバーまたはアプリケーションによるこのデータ・ソースへの書き込みアクティビティーのパーセンテージも判別できます。

```
書き込みアクティビティー =
  (INSERT ステートメント + UPDATE ステートメント +
  DELETE ステートメント) /
  (SELECT ステートメント + INSERT ステートメント +
  UPDATE ステートメント +
  DELETE ステートメント)
```

delete_time 削除応答時間

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時か、またはデータベース・モニター・カウンターの最後のリセット時のどちらか最近の方の時点以降に、このフェデレーテッド・サーバー・インスタンス上で実行されているすべてのアプリケーションまたは単一アプリケーションからの DELETE に対して、このデータ・ソースが応答に要した合計時間が含まれています (ミリ秒単位)。

応答時間とは、フェデレーテッド・サーバーが DELETE ステートメントをデータ・ソースにサブミットしてからデータ・ソースが DELETE を処理したことをフェデレーテッド・サーバーに応答するまでの時間です。

表 359. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	タイム・スタンプ
アプリケーション	appl_remote	タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、このデータ・ソースに対する DELETE が処

理されるのを待機するために生じた実際の時間を判別できます。この情報は、キャパシティー・プランニングおよびチューニングを行うときに便利です。

destination_service_class_id 宛先サービス・クラス ID : モニター・エレメント

このエレメントのしきい値違反レコードが生成された時に、アクティビティーが再マップされたサービス・サブクラスの ID。 しきい値アクションが REMAP ACTIVITY 以外の場合、このエレメントの値はゼロです。

エレメント ID

destination_service_class_id

エレメント・タイプ

情報

表 360. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-

使用法

このエレメントは、アクティビティーが再マップされたサービス・クラスをたどるのに使用できます。 このエレメントを使用して、特定のサービス・サブクラスに対してマップされたアクティビティー数の総計を計算することもできます。

diaglog_write_wait_time - 診断ログ・ファイル書き込み待機時間 : モニター・エレメント

db2diag ログ・ファイルへの書き込みの待機にかかった時間。値はミリ秒単位で示されます。

表 361. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 361. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 362. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このエレメントを使用すると、db2diag ログ・ファイルへの書き込みに費やされた時間が分かります。パーティション・データベース環境では、この時間の値が大きい場合、db2diag ログ・ファイルの競合を示している可能性があります (diagpath に共用ストレージが使用されている場合)。値が大きい場合は、過剰なロギングが行われている可能性もあります (例えば、すべての通知メッセージをログに記録するように **diaglevel** が設定されている場合)。

diaglog_writes_total - 診断ログ・ファイル書き込みの合計 : モニター・エレメント

エージェントが db2diag ログ・ファイルに書き込んだ回数。

表 363. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 363. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 364. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このエレメントを diaglog_write_wait_time と組み合わせて使用すると、db2diag ログ・ファイルへの書き込みに費やされた平均時間が分かります。

direct_read_reqs - 直接読み取り要求 : モニター・エレメント

1 つ以上のデータ・セクターの直接読み取り実行要求の数。

表 365. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 365. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 366. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 367. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
接続	event_conn	-
表スペース	event_tablespace	-

使用法

次の数式を使用すると、直接読み取りにより読み取られるセクターの平均数を計算できます。

```
direct_reads / direct_read_reqs
```

direct_read_time - 直接読み取り時間 : モニター・エレメント

直接読み取りの実行に要した経過時間。この値はミリ秒単位で示されます。

表 368. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 369. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 370. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
接続	event_conn	-
表スペース	event_tablespace	-

使用法

次の数式を使用して、セクター当たりの直接読み取り平均時間を計算します。

```
direct_read_time / direct_reads
```

平均時間が長い場合には、入出力が競合していることがあります。

direct_reads - データベースからの直接読み取り : モニター・エレメント

バッファー・プールを使用しない読み取り操作の数。

表 371. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 371. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 372. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 373. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
接続	event_conn	-
表スペース	event_tablespace	-

使用法

次の数式を使用すると、直接読み取りにより読み取られるセクターの平均数を計算できます。

```
direct_reads / direct_read_reqs
```

システム・モニターを使用して入出力を追跡するときにこのエレメントを利用すると、装置上のデータベース入出力とそれ以外の入出力を区別できます。

直接読み取りは、最小 512 バイト・セクター単位で処理されます。次の目的に使用します。

- LONG VARCHAR 列の読み取り
- LOB (ラージ・オブジェクト) 列の読み取り
- バックアップの実行

direct_write_reqs - 直接書き込み要求 : モニター・エレメント

1 つ以上のデータ・セクターの直接書き込み実行要求の数。

表 374. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 374. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 375. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 376. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
接続	event_conn	-
表スペース	event_tablespace	-

使用法

次の数式を使用すると、直接書き込みにより書き込まれるセクターの平均数を計算できます。

```
direct_writes / direct_write_reqs
```

direct_write_time - 直接書き込み時間 : モニター・エレメント

直接書き込みの実行に要した経過時間。この値はミリ秒単位で報告されます。

表 377. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE

表 377. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 378. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 379. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE

表 379. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
接続	event_conn	-
表スペース	event_tablespace	-

使用法

次の数式を使用して、セクター当たりの直接書き込み平均時間を計算します。

```
direct_write_time / direct_writes
```

平均時間が長い場合には、入出力が競合していることがあります。

direct_writes - データベースへの直接書き込み : モニター・エレメント

バッファー・プールを使用しない書き込み操作の数。

表 380. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 380. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 381. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 382. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
接続	event_conn	-
表スペース	event_tablespace	-

使用法

次の数式を使用すると、直接書き込みにより書き込まれるセクターの平均数を計算できます。

```
direct_writes / direct_write_reqs
```

システム・モニターを使用して入出力を追跡するときにこのエレメントを利用すると、装置上のデータベース入出力とそれ以外の入出力を区別できます。

直接書き込みは、最小 512 バイト・セクター単位で処理されます。次の目的に使用します。

- LONG VARCHAR 列の書き込み
- LOB (ラージ・オブジェクト) 列の書き込み
- リストアの実行
- ロードの実行
- MPFA が使用可能である場合 (デフォルト) の SMS 表スペースへの新規エクステントの割り振り

disconn_time データベース非活動化タイム・スタンプ

アプリケーションがデータベースとの接続を切断した日時 (データベース・レベルでは、アプリケーションが最後に切断した日時)。

エレメント ID

disconn_time

エレメント・タイプ

timestamp

表 383. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 このエレメントを使用すると、次の時点からの経過時間を計算できます。

- データベースがアクティブだったとき (データベース・レベルの情報の場合)。
- 接続がアクティブだったとき (接続レベルの情報の場合)。

disconnects 切断回数

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時か、またはデータベース・モニター・カウンターの最後のリセット時のどちらか最近の方の時点以降に、いずれかのアプリケーションに代わって、フェデレーテッド・サーバーがこのデータ・ソースから切断した合計回数が含まれています。

表 384. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

このエレメントを使用すると、フェデレーテッド・サーバーがいずれかのアプリケーションに代わってこのデータ・ソースから切断した合計回数を判別できます。このエレメントと CONNECT カウントを組み合わせて使用すると、データ・ソースに現在接続中であるとフェデレーテッド・サーバーのこのインスタンスが判断しているアプリケーションの数を判別できます。

dl_conns - デッドロックに関係している接続 : モニター・エレメント

デッドロックに関係している接続の数。

表 385. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
デッドロック[1]	event_deadlock	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

モニター・アプリケーションでこのエレメントを使用すると、イベント・モニター・データ・ストリーム内で処理されるデッドロック接続イベント・レコードの数を確認できます。

dynamic_sql_stmts 試行された動的 SQL ステートメント

試行された動的 SQL ステートメントの数。

表 386. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 387. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 このエレメントを使用すると、データベース・レベルまたはアプリケーション・レベルで成功した SQL ステートメントの合計数を計算できます。

```
  dynamic_sql_stmts
+ static_sql_stmts
- failed_sql_stmts
= モニター期間中のスループット
```

eff_stmt_text - 有効なステートメント・テキスト : モニター・エレメント

SQL ステートメントがステートメント・コンセントレーターの結果として変更された場合の、そのステートメントの有効なテキスト。

表 388. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
WLM_GET_ACTIVITY_DETAILS_COMPLETE 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

表 389. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitystmt	-

使用法

ステートメント・コンセントレーターが使用可能になっていて、ステートメント・コンセントレーターの結果としてステートメント・テキストが変更された場合、このモニター・エレメントには有効なステートメント・テキストが含まれます。そうでない場合は、このモニター・エレメントには 0 バイト長のテキスト・ストリングが含まれます。

effective_isolation - 有効な分離 : モニター・エレメント

このアクティビティーの有効な分離レベル。

表 390. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	常に収集される

表 391. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-

使用法

このエレメントを使用してアクティビティーの実行に使用された分離レベルを判別します。

effective_lock_timeout - 有効なロック・タイムアウト : モニター・エレメント

このアクティビティーの有効なロック・タイムアウト値。

表 392. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

effective_query_degree - 有効な照会の度合い : モニター・エレメント

このアクティビティーの有効な照会の並列処理の度合い。

表 393. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

表 394. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-

elapsed_exec_time ステートメント実行経過時間

DCS ステートメント・レベルでは、ホスト・データベース・サーバー上で SQL 要求の処理に要した経過時間を示します。この値はこのサーバーによって報告されています。 host_response_time エレメントと比較すると、このエレメントには DB2 Connect とホスト・データベース・サーバーの間のネットワーク経過時間が含まれていません。ほかのレベルでは、この値は特定のデータベースやアプリケーションのために実行されたすべてのステートメント、あるいは特定の回数のデータ転送を使用したステートメントについてのホスト実行時間の合計を示します。

表 395. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	ステートメント、タイム・スタンプ

表 395. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ステートメント、タイム・スタンプ
DCS データベース	dcs_dbase	ステートメント、タイム・スタンプ
DCS アプリケーション	dcs_appl	ステートメント、タイム・スタンプ
DCS ステートメント	dcs_stmt	ステートメント、タイム・スタンプ
データ伝送	stmt_transmissions	ステートメント、タイム・スタンプ

ステートメント・レベルでのスナップショット・モニターの場合、このカウンターはリセットできません。その他のレベルではこのカウンターはリセットできます。

使用法 このエレメントと経過時間に関する他のモニター・エレメントを組み合わせて使用すると、データベース・サーバーでの SQL 要求の処理を評価したり、パフォーマンス上の問題を特定するときに利用できます。

host_response_time エレメントからこのエレメントの値を引くと、 DB2 Connect とホスト・データベース・サーバーの間のネットワーク経過時間を計算できます。

注: dcs_dbase、dcs_appl、dcs_stmt、および stmt_transmissions レベルの場合、 *elapsed_exec_time element* は z/OS® データベースのみに適用されます。 DB2 Connect ゲートウェイが Windows、Linux、AIX、またはその他の UNIX データベースに接続している場合は、 *elapsed_exec_time* はゼロとして報告されます。

empty_pages_deleted - 削除された空ページ : モニター・エレメント

疑似空ページのすべてのキーは疑似削除されています。このモニター・エレメントは、削除された疑似空ページの数をレポートします。

表 396. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

empty_pages_reused - 再利用された空ページ : モニター・エレメント

疑似空ページのすべてのキーは疑似削除されています。このモニター・エレメントは、再利用された疑似空ページの数をレポートします。

表 397. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

entry_time - エントリー時間 : モニター・エレメント

アクティビティーが開始された時刻。

表 398. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

estimatedsqlcost_threshold_id - 見積もり SQL コストしきい値 ID : モニター・エレメント

アクティビティーに適用されていた ESTIMATEDSQLCOST しきい値の ID。

表 399. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、 ESTIMATEDSQLCOST しきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

estimatedsqlcost_threshold_value - 見積もり SQL コストしきい値 : モニター・エレメント

アクティビティーに適用されていた ESTIMATEDSQLCOST しきい値の上限。

表 400. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、ESTIMATEDSQLCOST しきい値がアクティビティーに適用されている場合、その値を判別します。

estimatedsqlcost_threshold_violated - 見積もり SQL コストしきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが ESTIMATEDSQLCOST しきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 401. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、アクティビティーに適用されていた ESTIMATEDSQLCOST しきい値にアクティビティーが違反したかどうかを判別します。

event_monitor_name イベント・モニター名

イベント・データ・ストリームを作成したイベント・モニターの名前。

エレメント ID
event_monitor_name

エレメント・タイプ
情報

表 402. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
イベント・ログ・ヘッダー	event_log_header	-

使用法 このエレメントを使用すると、分析中のデータとシステム・カタログ表内の特定のイベント・モニターを関連付けることができます。この名前は、SYSCAT.EVENTMONITORS カタログ表の NAME 列にある名前 (CREATE EVENT MONITOR および SET EVENT MONITOR のステートメントに指定されている) と同じものです。

event_time イベント時刻

イベントが発生した日時。

表 403. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表スペース	event_tablespace	-
表	event_table	-

使用法 このエレメントは、イベントを時系列順に関連付けるのに利用できます。

evmon_activates イベント・モニター活動化回数

イベント・モニターが活動化された回数。

エレメント ID

evmon_activates

エレメント・タイプ

カウンター

表 404. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
表	event_table	-
表スペース	event_tablespace	-
バッファー・プール	event_bufferpool	-
デッドロック	event_deadlock	-
デッドロック	event_dlconn	-
詳細付きデッドロック	event_detailed_dlconn	-

使用法 このエレメントを使用して、上記のイベント・タイプで戻された情報を関連付けます。このエレメントは、write-to-table イベント・モニターにのみ適用できます。このモニター・エレメントは、ファイルまたはパイプに書き込むイベント・モニター用には保持されていません。

一部のタイプの write-to-table イベント・モニターのみが evmon_activates モニター・エレメントを使用します (このエレメントを使用するイベント・モニター・タイプは、前述の「イベント・モニター情報」の表にリストされています)。それらのイベント・モニターは、活動化時に SYSCAT.EVENTMONITORS カタログ表の evmon_activates 列を更新します。この変更はログに記録されるため、DATABASE CONFIGURATION によって次が表示されます。

```
Database is consistent                                  = NO
```

イベント・モニターが AUTOSTART オプションを使用して作成されている場合、最初のユーザーがデータベースに接続してデータベースを非活動化するために即時に切断すると、ログ・ファイルが作成されます。

executable_id 実行可能 ID : モニター・エレメント

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、データ・サーバーで生成された不透明なバイナリー・トークン。 非 SQL アクティビティーの場合、長さが 0 のストリング値が戻されます。

表 405. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 406. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitystmt	-

使用法

このモニター・エレメントをさまざまなモニター・インターフェースに入力してセクションに関するデータを取得できます。 パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックを取得するのに使用される MON_GET_PKG_CACHE_STMT 表関数は、実行可能 ID を入力とみなします。

evmon_flushes イベント・モニター・フラッシュ回数

FLUSH EVENT MONITOR SQL ステートメントが発行された回数。

エレメント ID
evmon_flushes

エレメント・タイプ
情報

表 407. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
表	event_table	-
表スペース	event_tablespace	-
バッファー・プール	event_bufferpool	-

使用法 アプリケーションがデータベースに接続した後、データベース・マネージャーが FLUSH EVENT MONITOR SQL 要求を処理するごとに、この ID が増分します。このエレメントを使用すると、データベース、表、表スペース、およびバッファー・プール・データを特定できます。

execution_id ユーザー・ログイン ID

ユーザーがオペレーティング・システムにログインするときに指定した ID。この ID は、ユーザーがデータベースに接続するときに指定する auth_id とは異なります。

表 408. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
アプリケーション	appl	基本
DCS アプリケーション	dcs_appl_info	基本

表 409. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_connheader	-

使用法 このエレメントを使用すると、モニター対象のアプリケーションを実行している個人のオペレーティング・システム・ユーザー ID を識別できます。

failed_sql_stmts 失敗したステートメント操作

試行されたが失敗した SQL ステートメントの数。

表 410. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
データベース	dbase_remote	基本
アプリケーション	appl	基本
アプリケーション	appl_remote	基本
DCS データベース	dcs_dbase	基本
DCS アプリケーション	dcs_appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 411. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 このエレメントを使用すると、データベース・レベルまたはアプリケーション・レベルで成功した SQL ステートメントの合計数を計算できます。

```
  dynamic_sql_stmts
+ static_sql_stmts
- failed_sql_stmts
= モニター期間中のスループット
```

このカウントには、負の SQLCODE を受信したすべての SQL ステートメントを含みます。

このエレメントは、パフォーマンスが低い場合の原因の判別にも役に立ちます。これは、失敗したステートメントがあると、データベース・マネージャーで余分な時間がかかり、その結果としてデータベースのスループットが落ちるからです。

fcm_message_recv_volume - FCM メッセージ受信ボリューム : モニター・エレメント

FCM 通信層によって分散された内部要求 (RPC など) について受信されたデータの量。この値はバイト単位で示されます。

表 412. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 413. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このエレメントを使用して、FCM サブシステムを介して送信されているデータ・ボリュームのうちのどれほどの量が、実際の表データではなく、要求メッセージまた

は応答メッセージのトラフィックのボリュームであるかを判別します。

fcm_message_recv_wait_time - FCM メッセージの受信待機時間 : モニター・エレメント

以前に送信された FCM 要求メッセージの結果を含む FCM 応答メッセージの待機にエージェントが費やした時間。この時間は、他方の側で要求メッセージを処理するのに必要な時間と、パーティション間で FCM を使用して応答を送信するのに必要な時間の両方を反映します。値はミリ秒単位で示されます。

表 414. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 415. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このエレメントを使用すると、複数パーティション・インスタンスにおいて、特定のパーティション上で、他のパーティション上で要求が処理されるのを待機するのにどれだけの時間がかかったかを判別できます。

fcm_message_recvs_total - FCM メッセージ受信の合計 : モニター・エレメント

以前に送信された FCM 要求メッセージの結果を含む FCM 応答メッセージの一部として受信されるバッファーの合計数。

表 416. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用すると、受信された FCM メッセージごとの平均ボリュームと、単一の FCM メッセージの受信を待機するのにかかった平均時間の両方を判別できます。

fcm_message_send_volume - FCM メッセージ送信ボリューム : モニター・エレメント

内部 FCM 要求を介して送信されるデータ・ボリュームの量。この値はバイト単位で示されます。

表 417. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 417. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、FCM サブシステムを介して送信されているデータ・ボリュームのうちのどれほどの量が、実際の表データを送信するためにではなく、要求メッセージおよび応答メッセージのトラフィックを送信するために使用されているかを判別します。

fcm_message_send_wait_time - FCM メッセージの送信待機時間 : モニター・エレメント

FCM メッセージ送信に対するブロックにかかった時間。値はミリ秒単位で示されます。このモニター・エレメントは、データベース・システム上で内部要求を配布するときに、FCM チャネルから FCM バッファーがフラッシュされるためのブロックに費やした時間を反映します。

表 418. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

使用法

このエレメントを使用して、FCM サブシステムを介した FCM 要求メッセージの送信を待機するのにエージェントが費やしている時間を判別します。 FCM デーモンのビジー状況によっては、エージェントはメッセージの送信を試行する際に待機しなければならない場合があります。

fcm_message_sends_total - FCM メッセージ送信の合計 : モニター・エレメント

FCM 通信メカニズムを使用して、内部要求の一部として配布されたバッファーの合計数。

表 419. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 420. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このエレメントを使用して、FCM 要求メッセージごとに送信されたデータの平均の量と、FCM メッセージごとの平均の待機時間の両方を判別します。

fcm_recv_volume - FCM 受信ボリューム : モニター・エレメント

FCM 通信層を介して受信されたデータの合計量。この値はバイト単位で示されます。

表 421. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 422. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このパーティション上で FCM を使用して受信されたデータ (メッセージ・トラフィックと表キュー・データの両方を含む) の合計ボリュームを示します。

fcm_recv_wait_time - FCM 受信待機時間 : モニター・エレメント

FCM を介したデータの受信の待機にかかった合計時間。値はミリ秒単位で示されます。

表 423. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 424. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE

表 424. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	system_metrics 文書に報告されます。	-

使用法

このエレメントを使用して、このデータベース・パーティション上で FCM を介したデータの受信の待機にかかった合計時間を判別します。これには、要求メッセージに対する応答からのデータと、表キュー・データの両方が含まれます。

fcm_recvs_total - FCM 受信の合計 : モニター・エレメント

FCM 通信メカニズムを使用して、内部要求のために受信されたバッファーの合計数。fcm_recvs_total モニター・エレメントの値は、fcm_message_recvs_total モニター・エレメントと fcm_tq_recvs_total モニター・エレメントの値の合計です。

表 425. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 426. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このエレメントを **fcm_recv_wait_time** モニター・エレメントと一緒に使用して、FCM 受信操作ごとの平均待機時間と、FCM 受信操作から戻される平均ボリュームを判別します。

fcm_send_volume - FCM 送信ボリューム : モニター・エレメント

FCM 通信層によって配布されたデータの合計量。この値はバイト単位で示されます。

表 427. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 427. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

使用法

このモニター・エレメントを使用して、FCM を使用して送信されたデータ (メッセージ・トラフィックと表キュー・データの両方を含む) の合計量を判別します。

fcm_send_wait_time - FCM 送信待機時間 : モニター・エレメント

FCM 送信操作に対するブロックにかかった時間。これには、内部要求用のバッファーがフラッシュされるのを待機するのに要した時間と、表キューでデータを送信するとき、ウィンドウ・カウントの確認応答を待機するのに要した時間が含まれます。値はミリ秒単位で示されます。

表 428. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 428. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 429. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このエレメントを使用して、FCM を介したデータ送信の待機にかかった合計時間を判別します。これには、要求メッセージと表キュー・データの両方が含まれます。

fcm_sends_total - FCM 送信の合計 : モニター・エレメント

内部 FCM 通信層を使用して送信されたバッファーの合計数。

表 430. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE

表 430. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 431. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このエレメントを使用して、FCM 受信操作ごとの平均待機時間と、FCM 受信操作から戻される平均ボリュームを判別します。

fcm_tq_recv_volume - FCM 表キュー受信ボリューム : モニター・エレメント

FCM 通信層によって表キュー上で受信されたデータの量。この値はバイト単位で示されます。

表 432. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 432. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 433. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このモニター・エレメントを使用して、表キューを使用して受信されたデータの合計ボリュームを判別します。

fcm_tq_recv_wait_time - FCM 表キュー受信待機時間 : モニター・エレメント

表キューから次のバッファーを受信するのを待機していた時間。値はミリ秒単位で示されます。

表 434. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 434. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 435. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このエレメントを使用して、表キュー上のデータを受信するのをエージェントが待機している時間を判別します。

fcm_tq_recvs_total - FCM 表キュー受信の合計 : モニター・エレメント

内部 FCM 通信メカニズムを使用して表キューから受信されたバッファーの合計数。

表 436. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 437. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このエレメントを **fcm_tq_recv_volume** および **fcm_tq_recv_wait_time** と組み合わせて使用すると、受信した表キュー・バッファーごとの平均待機時間とボリュームを判別できます。

fcm_tq_send_volume - FCM 表キュー送信ボリューム : モニター・エレメント

FCM 通信層によって表キュー上で送信されたデータの量。この値はバイト単位で示されます。

表 438. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 439. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE

表 439. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このモニター・エレメントを使用して、FCM を使用して表キュー・バッファー送信によって送信されたデータの合計ボリュームを判別します。

fcm_tq_send_wait_time - FCM 表キュー送信待機時間 : モニター・エレメント

表キューによって次のバッファーを送信するのを待機していた時間。これは、表キューの受信側の終端からのウィンドウ・カウントの確認応答の待機にかかった時間を反映します。値はミリ秒単位で示されます。

表 440. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 441. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE

表 441. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このモニター・エレメントを使用して、FCM を使用した、表キューによるデータ・バッファーの送信を待機するのにかかっている時間を判別します。

fcm_tq_sends_total - FCM 表キュー送信の合計 : モニター・エレメント

内部 FCM 通信メカニズムを使用して送信された表キュー・データが入っているバッファーの合計数。

表 442. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 443. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このエレメントを **fcm_tq_send_volume** および **fcm_tq_send_wait_time** モニター・エレメントと組み合わせて使用すると、平均データ・ボリューム、および表キューを使用してバッファーが送信されるのを待機した時間を判別できます。

fetch_count 成功したフェッチの数

成功した物理フェッチの数か、または試みられた物理フェッチの数。スナップショット・モニター・レベルに応じて決まります。

- stmt および dynsql スナップショット・モニター・レベルでありイベント・タイプがステートメントの場合は、特定のカーソル上で実行されて成功したフェッチの数。
- dcs_stmt スナップショット・モニター・レベルの場合は、 (アプリケーションによってフェッチされた行数にかかわらず) ステートメントの実行時に試みられた物理フェッチの数。この状態の場合、**fetch_count** は、ステートメントの処理中にサーバーがゲートウェイに対して応答データを送り返す必要があった回数を表します。

表 444. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント
DCS ステートメント	dcs_stmt	ステートメント
動的 SQL	dynsql	ステートメント

動的 SQL スナップショット・モニターの場合、このカウンターはリセットできます。

表 445. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_stmt	-

使用法

このエレメントを使用すると、データベース・マネージャー内の現在のアクティビティーのレベルがわかります。

ステートメント・イベント・モニターは、パフォーマンス上の理由から、すべての FETCH ステートメントを対象にステートメント・イベント・レコードを生成するわけではありません。レコード・イベントが生成されるのは、FETCH がゼロ以外の SQLCODE を戻した場合だけです。

files_closed - クローズしたデータベース・ファイル : モニター・エレメント

閉じられたデータベース・ファイルの合計数。

表 446. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 447. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 448. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
表スペース	event_tablespace	-

使用法

データベース・マネージャーは、バッファー・プールへの書き込みおよびバッファー・プールからの読み取りを行うためにファイルを開きます。アプリケーションが同時に開けるデータベース・ファイルの最大数は、 **maxfilop** 構成パラメーターによりコントロールされています。最大値に達すると、新しいファイルを開く前に、ファイルが 1 つ閉じられます。実際に開かれたファイルの数と閉じられたファイルの数は等しくならないことに注意してください。

このエレメントを使用すると、**maxfilop** 構成パラメーターの最適な値を判別するのに役立てることができます。

first_active_log 先頭アクティブ・ログ・ファイル番号

最初のアクティブ・ログ・ファイルのファイル番号。

エレメント ID

first_active_log

エレメント・タイプ

情報

表 449. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	detail_log	基本

表 450. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 このエレメントと *last_active_log* および *current_active_log* エレメントを組み合わせて使用すると、アクティブ・ログ・ファイルの範囲を判別できます。アクティブ・ログ・ファイルの範囲を知っていると、ログ・ファイルに必要なディスク・スペースを判別するのに役立ちます。

また、このエレメントを使用すると、どのログ・ファイルにデータがあるか判別でき、スプリット・ミラー・サポートが必要なログ・ファイルを識別するのに役立ちます。

first_overflow_time 最初のイベント・オーバーフロー時刻

このオーバーフロー・レコードに記録されている最初のオーバーフローの日時。

表 451. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
オーバーフロー・レコード	event_overflow	-

使用法 このエレメントと *last_over_flow time* を組み合わせて使用すると、オーバーフロー・レコードを生成するのに要した経過時間を計算できます。

fs_caching - ファイル・システム・キャッシング : モニター・エレメント

特定の表スペースがファイル・システム・キャッシングを使用するかどうかを示します。**fs_caching** が 0 の場合、ファイル・システム・キャッシングは使用可能です。 **fs_caching** が 1 の場合は、ファイル・システム・キャッシングは使用不可です。

表 452. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 453. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本

表 454. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表スペース	event_tablespace	-

fs_id - 固有のファイル・システム識別番号 : モニター・エレメント

このエレメントは、ストレージ・パスまたはコンテナーが指し示すファイル・システムに対してオペレーティング・システムが提供する固有の識別番号を示します。

表 455. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE

表 456. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	db_sto_path_info	バッファー・プール

使用法

このエレメントを次のエレメントと共に使用して、データベースのスペース使用率に関するデータを収集します。

- **db_storage_path**
- **sto_path_free_sz**
- **fs_used_size**
- **fs_total_size**
- **fs_type**

fs_total_size - ファイル・システムの合計サイズ : モニター・エレメント

このエレメントは、ストレージ・パスまたはコンテナーが指し示すファイル・システムの容量を示します。

表 457. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE

表 458. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	db_sto_path_info	バッファー・プール

使用法

このエレメントを次のエレメントと共に使用して、データベースのスペース使用率に関するデータを収集することができます。

- **db_storage_path**
- **sto_path_free_sz**

- **fs_used_size**
- **fs_id**
- **fs_type**

fs_type ファイル・システム・タイプ

このエレメントは、ストレージ・パスが指し示すファイル・システムのタイプを示します。このファイル・システム・タイプはオペレーティング・システムによって提供されます。

エレメント ID

fs_type

エレメント・タイプ

情報

表 459. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	db_sto_path_info	バッファー・プール

使用法 このエレメントを次のエレメントと共に使用して、データベースのスペース使用率に関するデータを収集することができます。

- db_storage_path
- sto_path_free_sz
- fs_used_size
- fs_total_size
- fs_id

fs_used_size - ファイル・システム上で使用されるスペースの量 : モニター・エレメント

このエレメントは、ストレージ・パスまたはコンテナーが指し示すファイル・システム上で既に使用されているスペースの量を示します。

表 460. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE

表 461. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	db_sto_path_info	バッファー・プール

使用法

このエレメントを次のエレメントと共に使用して、データベースのスペース使用率に関するデータを収集することができます。

- **db_storage_path**
- **sto_path_free_sz**
- **fs_total_size**
- **fs_id**
- **fs_type**

gw_comm_error_time 通信エラー時刻

DCS アプリケーションがホスト・データベースへの接続を試みたときに、または SQL ステートメントを処理していたときに通信エラー (SQL30081) が最後に発生した日時。

表 462. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	タイム・スタンプ

使用法

このエレメントは、通信エラーおよび管理用通知ログ に記録される通信エラー・ログと組み合わせて、問題判別に使用できます。

gw_comm_errors 通信エラー

DCS アプリケーションがホスト・データベースへの接続を試みたときに、または SQL ステートメントを処理していたときに通信エラー (SQL30081) が発生した回数。

表 463. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 時間をかけて通信エラーの数をモニターすることによって、DB2 Connect ゲートウェイに特定のホスト・データベースとの接続の問題があるかどうかを評価できます。通常のエラーしきい値と考えられるものを設定できるため、エラーの数がこのしきい値を超えるたびに、通信エラーの調査を行うことができます。

管理用通知ログ に記録される通信エラー・ログとともに、このエレメントを問題判別に使用してください。

gw_con_time DB2 Connect ゲートウェイの最初の接続開始

ホスト・データベースへの最初の接続が DB2 Connect ゲートウェイから開始された日時。

表 464. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	タイム・スタンプ
DCS アプリケーション	dcs_appl	タイム・スタンプ

使用法 このエレメントは、DCS アプリケーションに関する問題判別に使用します。

gw_connections_top ホスト・データベースへの同時接続の最大数

最初のデータベース接続以降、 DB2 Connect ゲートウェイが処理したホスト・データベースへの同時接続の最大数。

エレメント ID

gw_connections_top

エレメント・タイプ

水準点

表 465. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	基本

使用法 このエレメントは、 DB2 Connect ゲートウェイでのアクティビティーのレベルと、関連したシステム・リソースの使用状況を知るのに役立ちます。

gw_cons_wait_client クライアントの要求送信を待機している接続の数

DB2 Connect ゲートウェイが処理しているホスト・データベースへの接続のうち、クライアントが要求を送信するのを待機している現在の接続数。

エレメント ID

gw_cons_wait_client

エレメント・タイプ

ゲージ

表 466. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本
DCS データベース	dcs_dbase	基本

使用法 この値は頻繁に変わることがあります。実際のゲートウェイの使用率を把握するには、長期にわたって周期的にサンプリングを行わなければなりません。

gw_cons_wait_host ホストの応答を待機している接続の数

DB2 Connect ゲートウェイが処理しているホスト・データベースへの接続のうち、ホストからの応答を待機している現在の接続数。

エレメント ID

gw_cons_wait_host

エレメント・タイプ

ゲージ

表 467. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本
DCS データベース	dcs_dbase	基本

使用法 この値は頻繁に変わることがあります。実際のゲートウェイの使用率を把握するには、長期にわたって周期的にサンプリングを行わなければなりません。

gw_cur_cons DB2 Connect の現在の接続数

DB2 Connect ゲートウェイが処理しているホスト・データベースへの現在の接続数。

表 468. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本
DCS データベース	dcs_dbase	基本

使用法 このエレメントは、 DB2 Connect ゲートウェイでのアクティビティーのレベルと、関連したシステム・リソースの使用状況を知るのに役立ちます。

gw_db_alias ゲートウェイでのデータベース別名

ホスト・データベースに接続するために DB2 Connect ゲートウェイで使用される別名。

エレメント ID

gw_db_alias

エレメント・タイプ

情報

表 469. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl_info	基本

使用法 このエレメントは、DCS アプリケーションに関する問題判別に使用します。

gw_exec_time DB2 Connect ゲートウェイ処理の経過時間

DB2 Connect ゲートウェイがアプリケーションの要求を処理するのに要した時間 (接続が確立された以降)、または単一ステートメントを処理するのに要した時間 (秒およびマイクロ秒単位)。

表 470. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl	ステートメント、タイム・スタンプ
DCS ステートメント	dcs_stmt	ステートメント、タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用して、全体の処理時間のうちどの部分が DB2 Connect ゲートウェイの処理によるものかを判別します。

gw_total_cons DB2 Connect の接続試行合計回数

最後の db2start コマンドまたは最後のリセット以降に、 DB2 Connect ゲートウェイから試行された接続の合計回数。

表 471. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本
DCS データベース	dcs_dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントは、 DB2 Connect ゲートウェイでのアクティビティーのレベルと、関連したシステム・リソースの使用状況を知るのに役立ちます。

hadr_connect_status HADR 接続状況 : モニター・エレメント

データベースの高可用性災害時リカバリー (HADR) の現在の接続状況。

表 472. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、データベースの HADR 接続状況を判別できます。

このエレメントのデータ・タイプは整数です。

データベースが HADR 1 次またはスタンバイの役割の場合は、このエレメントの値は、以下のいずれかの定数です。

SQLM_HADR_CONN_CONNECTED
データベースはそのパートナー・ノードに接続しています。

SQLM_HADR_CONN_DISCONNECTED
データベースはそのパートナー・ノードに接続していません。

SQLM_HADR_CONN_CONGESTED
データベースはそのパートナー・ノードに接続していますが、接続が混雑しています。1 次とスタンバイのペアの間に TCP/IP ソケット接続が依然として存在しているものの、一方の終端が他方の終端に送信できない場合に、接続は混雑します。例えば、受信側の終端がソケット接続から受信していないと、 TCP/IP 送信スペースがいっぱいになります。ネットワーク接続が混雑する理由には、次のものがあります。

- ネットワークを共有するリソースが多すぎるか、ネットワークが 1 次 HADR ノードのトランザクション・ボリュームにとって低速である。
- スタンバイ HADR ノードのあるサーバーが強力でないので、必要な速度で通信サブシステムから情報を取り出せない。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_connect_time HADR 接続時刻 : モニター・エレメント

高可用性災害時回復 (HADR) 接続時刻、HADR 輻輳 (ふくそう) 時刻、または HADR 切断時刻のいずれかを示します。

表 473. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用すると、現在の HADR 接続状況が始まった時刻を判別できます。

データベースが HADR 1 次またはスタンバイの役割の場合は、このエレメントの意味は、 **hadr_connect_status** エレメントの値に従属します。

- **hadr_connect_status** エレメントの値が SQLM_HADR_CONN_CONNECTED の場合は、このエレメントは接続時刻を示す。
- **hadr_connect_status** エレメントの値が SQLM_HADR_CONN_CONGESTED の場合は、このエレメントは輻輳 (ふくそう) の開始時刻を示す。
- **hadr_connect_status** エレメントの値が SQLM_HADR_CONN_DISCONNECTED の場合は、このエレメントは切断時刻を示す。

HADR エンジン・ディスパッチ可能単位 (EDU) の開始以降に接続が行われていない場合、接続状況は「切断」として報告され、切断時刻に HADR EDU 起動時刻が使用されます。 HADR 接続イベントや切断イベントは比較的頻度が低いので、DFT_MON_TIMESTAMP スイッチが OFF の場合でも時刻は収集されて報告されます。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_heartbeat HADR ハートビート : モニター・エレメント

高可用性災害時回復 (HADR) 接続上で欠落しているハートビートの数。データベースが HADR 1 次またはスタンバイの役割の場合は、このエレメントは HADR 接続の正常性を示します。

表 474. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

スナップショット・モニターの場合、このカウンターはリセットできません。

使用上の注意:

このエレメントを使用して、HADR 接続の正常性を判別できます。

ハートビートとは、他の HADR データベースから定期的に送信されるメッセージのことです。このエレメントの値がゼロの場合は、ハートビートは欠落しておらず、接続は健全です。値が大きくなるほど、接続の状態は悪くなります。

切断モードでは、欠落したハートビートは適用されないため、常に 0 として表示されます。

HADR データベースは、 HADR_TIMEOUT データベース構成パラメーターで定義されている時間間隔の 4 分の 1 か、または 30 秒のうちの短い方の時間内に、他のデータベースからの 1 つ以上のハートビート信号を待ちます。例えば、HADR_TIMEOUT 値が 80 (秒) の場合は、 HADR データベースは 20 秒ごとに他のデータベースからの 1 つ以上のハートビート信号を待ちます。

このエレメントのデータ・タイプは整数です。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_local_host - HADR ローカル・ホスト : モニター・エレメント

高可用性災害時リカバリー (HADR) ローカル・ホスト名値は、ホスト名のストリングか、「1.2.3.4」などの IP アドレスのストリングとして表示されます。

表 475. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、有効な HADR ローカル・ホスト名を判別できます。HADR データベース構成パラメーターは静的です。パラメーターに対する変更内容は、データベースが停止して再始動するまでは有効になりません。このモニター・エレメントは、データベース構成ファイル中の値ではなく、 HADR システムが実際に使用している値を報告します。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

注: 使用する名前は、1 つの IP アドレスに解決されなければなりません。複数のアドレスに解決される名前を使用すると、HADR を始動しようとするときにエラーが発生します。

hadr_local_service HADR ローカル・サービス : モニター・エレメント

ローカル HADR TCP サービス。この値は、サービス名のストリングか、ポート番号のストリングとして表示されます。

表 476. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、有効な HADR ローカル・サービス名を判別できます。 HADR データベース構成パラメーターは静的です。パラメーターに対する変更内容は、データベースが停止して再始動するまでは有効になりません。このモニター・エレメントは、データベース構成ファイル中の値ではなく、 HADR システムが実際に使用している値を報告します。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_log_gap HADR ログ・ギャップ

このエレメントは、 1 次ログ・シーケンス番号 (LSN) とスタンバイ・ログ LSN の間のギャップの現行の平均を示します。ギャップはバイト数で測定されます。

表 477. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、 1 次 HADR データベース・ログとスタンバイ HADR データベース・ログの間のギャップを判別できます。

ログ・ファイルが切り捨てられると、次のログ・ファイルの LSN は、直前のファイルが切り捨てられていないかのように始まります。この LSN の穴には、ログ・データは含まれません。この種の穴により、ログ・ギャップが 1 次 HADR データベース・ログとスタンバイ HADR データベース・ログの間の実際のログの差を反映しない可能性があります。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_peer_window HADR ピア・ウィンドウ : モニター・エレメント

HADR_PEER_WINDOW データベース構成パラメーターの値。

表 478. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用すると、HADR_PEER_WINDOW データベース構成パラメーターの値を判別できます。

hadr_peer_window_end HADR ピア・ウィンドウ終了 : モニター・エレメント

高可用性災害時リカバリー (HADR) 1 次データベースがアクティブである限り、この 1 次データベースがピアまたは切断済みピア状態であることを保証する期間が終了する時点。

表 479. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用すると、1 次データベースがピアまたは切断済みピア状態であることを保証する期間が終了する時点を特定することができます。

1 次データベースによって報告される値と、スタンバイ・データベースによって報告される値が違うことがあります。この違いが生じる理由は、1 次データベースはハートビート・メッセージを送信する際に値を更新しますが、この新しい値がスタンバイ・データベースで表示されるのはこのメッセージがスタンバイ・データベースで受け取られて処理された後に限られるからです。

データベースがピアまたは切断済みピア状態でなくなっても、このモニター・エレメントの値はリセットされません。最後に認識された値が保持され、戻されます。データベースがピア状態に達することがなかった場合は、ゼロの値が戻されます。

ピア・ウィンドウ終了時刻は、1 次データベースによって設定された後、スタンバイ・データベースに送信されます。このため、ピア・ウィンドウ終了時刻の値は 1 次データベースのクロックに基づいています。ピア・ウィンドウ終了時刻と 1 次データベースのダウン時刻を比較する際に、2 つのクロックの同期が十分取れていない場合には、オフセットを追加してタイム・スタンプを 1 次データベースのクロックに変換する必要が生じることがあります。

hadr_primary_log_file HADR 1 次ログ・ファイル : モニター・エレメント

1 次 HADR データベース上の現行ログ・ファイルの名前。

表 480. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、1 次 HADR データベース上の現行ログ・ファイルを判別できます。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_primary_log_lsn HADR 1 次ログ LSN : モニター・エレメント

1 次 HADR データベースの現在のログの位置。ログ・シーケンス番号 (LSN) とは、データベースのログ・ストリーム中のバイト・オフセットのことです。

表 481. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、1 次 HADR データベース上の現在のログの位置を判別できます。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_primary_log_page HADR 1 次ログ・ページ : モニター・エレメント

1 次 HADR データベース上の現在のログ位置を示す現行ログ・ファイルのページ番号。ページ番号はログ・ファイルに関連しています。例えば、ページ・ゼロはファイルの先頭です。

表 482. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、1 次 HADR データベース上の現行ログ・ページを判別できます。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_remote_host HADR リモート・ホスト : モニター・エレメント

高可用性災害時リカバリー (HADR) リモート・ホスト名値は、ホスト名のストリングか、「1.2.3.4」などの IP アドレスのストリングとして表示されます。

表 483. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、有効な HADR リモート・ホスト名を判別できます。HADR データベース構成パラメーターは静的です。パラメーターに対する変更内容は、データベースが停止して再始動するまでは有効になりません。このモニター・エレメントは、データベース構成ファイル中の値ではなく、 HADR システムが実際に使用している値を報告します。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。*hadr_role* モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

注: 使用する名前は、1 つの IP アドレスに解決されなければなりません。複数のアドレスに解決される名前を使用すると、HADR を始動しようとするときにエラーが発生します。

hadr_remote_instance HADR リモート・インスタンス : モニター・エレメント

リモート HADR インスタンス名。

表 484. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、有効な HADR リモート・インスタンス名を判別できます。 HADR データベース構成パラメーターは静的です。パラメーターに対する変更内容は、データベースが停止して再始動するまでは有効になりません。このモニター・エレメントは、データベース構成ファイル中の値ではなく、 HADR システムが実際に使用している値を報告します。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。 **hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_remote_service HADR リモート・サービス : モニター・エレメント

リモート HADR TCP サービス。この値は、サービス名のストリングか、ポート番号のストリングとして表示されます。

表 485. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、有効な HADR リモート・サービス名を判別できます。 HADR データベース構成パラメーターは静的です。パラメーターに対する変更内容は、データベースが停止して再始動するまでは有効になりません。このモニター・エレメントは、データベース構成ファイル中の値ではなく、 HADR システムが実際に使用している値を報告します。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。 **hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_role HADR の役割

データベースの高可用性災害時リカバリー (HADR) の現在の役割。

表 486. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、データベースの HADR の役割を判別できます。

このエレメントのデータ・タイプは整数です。

このエレメントの値は、以下のいずれかの定数です。

SQLM_HADR_ROLE_STANDARD
データベースは HADR データベースではありません。

SQLM_HADR_ROLE_PRIMARY
データベースは 1 次 HADR データベースです。

SQLM_HADR_ROLE_STANDBY
データベースはスタンバイ HADR データベースです。

hadr_standby_log_file HADR スタンバイ・ログ・ファイル : モニター・エレメント

スタンバイ HADR データベース上の現行ログ・ファイルの名前。

表 487. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、スタンバイ HADR データベース上の現行ログ・ファイルを判別できます。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_standby_log_lsn HADR スタンバイ・ログ LSN : モニター・エレメント

スタンバイ HADR データベースの現在のログの位置。ログ・シーケンス番号 (LSN) とは、データベースのログ・ストリーム中のバイト・オフセットのことです。

表 488. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、スタンバイ HADR データベース上の現在のログの位置を判別できます。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_standby_log_page HADR スタンバイ・ログ・ページ : モニター・エレメント

スタンバイ HADR データベース上の現在のログ位置を示す現行ログ・ファイルのページ番号。ページ番号はログ・ファイルに関連しています。例えば、ページ・ゼロはファイルの先頭です。

表 489. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、スタンバイ HADR データベース上の現行ログ・ページを判別できます。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_state HADR の状態 : モニター・エレメント

データベースの高可用性災害時リカバリー (HADR) の現在の状態。

表 490. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、データベースの HADR の状態を判別できます。

このエレメントのデータ・タイプは整数です。データベースが HADR 1 次またはスタンバイの役割の場合は、このエレメントの値は、以下のいずれかの定数です。

SQLM_HADR_STATE_DISCONNECTED
データベースはそのパートナー・データベースに接続していません。

SQLM_HADR_STATE_LOC_CATCHUP
データベースはローカル・キャッチアップを行っています。

SQLM_HADR_STATE_REM_CATCH_PEND
データベースはそのパートナーに接続してリモート・キャッチアップを行うのを待っています。

SQLM_HADR_STATE_REM_CATCHUP
データベースはリモート・キャッチアップを行っています。

SQLM_HADR_STATE_PEER
1 次データベースとスタンバイ・データベースが接続され、ピア状態になっています。

SQLM_HADR_STATE_DISCONN_PEER
1 次データベースとスタンバイ・データベースが切断済みピア状態になっています。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_syncmode HADR 同期モード : モニター・エレメント

データベースの高可用性災害時リカバリー (HADR) の現在の同期モード。

表 491. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、データベースの HADR 同期モードを判別できます。

このエレメントのデータ・タイプは整数です。

HADR データベース構成パラメーターは静的です。パラメーターに対する変更内容は、データベースが停止して再始動するまでは有効になりません。このモニター・エレメントは、データベース構成ファイル中の値ではなく、 HADR システムが実際に使用している値を報告します。

データベースが HADR 1 次またはスタンバイの役割の場合は、このエレメントの値は、以下のいずれかの定数です。

SQLM_HADR_SYNCMODE_SYNC
同期モード。

SQLM_HADR_SYNCMODE_NEARSYNC
近似同期モード。

SQLM_HADR_SYNCMODE_ASYNC
非同期モード。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hadr_timeout HADR タイムアウト : モニター・エレメント

そのパートナーからの通信がなく、パートナーとの間の接続が失敗したと HADR データベース・サーバーが見なすまでに要する秒数。

表 492. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	hadr	基本

使用法

このエレメントを使用して、有効な HADR タイムアウト値を判別できます。HADR データベース構成パラメーターは静的です。パラメーターに対する変更内容は、データベースが停止して再始動するまでは有効になりません。このモニター・エレメントは、データベース構成ファイル中の値ではなく、 HADR システムが実際に使用している値を報告します。

データベースの HADR の役割が標準の場合は、このエレメントは無視されます。**hadr_role** モニター・エレメントを使用して、データベースの HADR の役割を判別できます。

hash_join_overflows ハッシュ結合のオーバーフロー

ハッシュ結合データが、使用可能なソート・ヒープ・スペースを超えた回数。

エレメント ID
hash_join_overflows

エレメント・タイプ
カウンター

表 493. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 494. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 データベース・レベルでは、 hash_join_small_overflows の値がこの hash_join_overflows の 10% を超える場合は、ソート・ヒープ・サイズを大きくすることを検討してください。アプリケーション・レベルの値は、個々のアプリケーションについてハッシュ結合のパフォーマンスを評価するときに使用できます。

hash_join_small_overflows ハッシュ結合の短精度オーバーフロー

ハッシュ結合データが、使用可能なソート・ヒープ・スペースを 10% を超えない範囲で超えた回数。

エレメント ID

hash_join_small_overflows

エレメント・タイプ

カウンター

表 495. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 496. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 この値と hash_join_overflows の値が大きい場合は、ソート・ヒープのしきい値を大きくすることを検討してください。この値が hash_join_overflows の 10% を超える場合は、ソート・ヒープ・サイズを大きくすることを検討してください。

histogram_type ヒストグラム・タイプ : モニター・エレメント

ヒストグラムのタイプ (ストリング形式)。

ヒストグラムには、6 つのタイプがあります。

CoordActQueueTime

ネストなしアクティビティーがキュー (しきい値キューなど) に入れられている時間のヒストグラム。コーディネーター・パーティション上で測定されます。

CoordActExecTime

ネストなしアクティビティーがコーディネーター・パーティションで実行されている時間のヒストグラム。実行時間には、初期化されている時間またはキューに入れられている時間は含まれません。カーソルの場合、実行時間にはオープン、フェッチ、およびクローズ要求に要する時間のみ含まれます。アクティビティーがサービス・サブクラス間で再マップされると、実行時間ヒストグラムは、アクティビティーが実行を完了するサービス・サブクラスについてのみ更新されます。

CoordActLifetime

ネストなしアクティビティーがデータベース・マネージャーによって識別されてから、そのアクティビティーが実行を完了するまでの経過時間のヒストグラム。コーディネーター・パーティション上で測定されます。アクティビティーをサービス・サブクラス間で再マップした場合、存続時間ヒストグラムは、アクティビティーが実行を完了するサービス・サブクラスについてのみ更新されます。

CoordActInterArrivalTime

ネストなしコーディネーター・アクティビティーの到着から次の到着までの間の時間間隔のヒストグラム。到着間隔時間の平均値は、アクティビティーがシステムに入るときに使用されるサービス・サブクラスを対象に計算されます。アクティビティーをサービス・サブクラス間で再マップした場合、アクティビティーの再マップ先のサービス・サブクラスの到着間隔時間ヒストグラムは影響を受けません。

CoordActEstCost

ネストなし DML アクティビティーの見積コストのヒストグラム。アクティビティーの見積コストは、アクティビティーがシステムに入るときのサービス・サブクラスに関してのみカウントされます。

ReqExecTime

要求の実行時間のヒストグラム。要求には、コーディネーター・パーティションでの要求と、コーディネーター・パーティション、非コーディネーター・パーティションの両方でのサブリクエスト (RPC 要求、SMP サブエージェント要求など) が含まれます。含まれている要求には、アクティビティーが関連付けられている場合もあれば、そうでない場合もあります。例えば、このヒストグラムには PREPARE 要求と OPEN 要求の両方が含まれていますが、OPEN 要求の方は常にカーソル・アクティビティーに関連付けられているのに対し、PREPARE 要求の方はアクティビティーの一部ではありません。再マップに関係するサービス・サブクラスの実行時間ヒストグラムでは、そのサービス・サブクラス内で部分要求が費やす実行時間部分がカウントされます。

表 497. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_histogrambin	-

使用法

このエレメントを使用すると、ヒストグラムのタイプを識別できます。複数のヒストグラムが同じ統計レコードに所属する場合がありますが、各タイプごとに 1 つずつしか所属しません。

host_ccsid ホスト・コード化文字セット ID

ホスト・データベースのコード化文字セット ID (CCSID) です。

エレメント ID

host_ccsid

エレメント・タイプ

情報

表 498. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl_info	基本

使用法 このエレメントは、DCS アプリケーションに関する問題判別に使用します。

host_db_name ホスト・データベース名

情報が収集されているホスト・データベースの実名、またはアプリケーションの接続先のホスト・データベースの実名。これは、このホスト・データベースが作成されたときに付けられた名前です。

表 499. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	基本
DCS アプリケーション	dcs_appl_info	基本

使用法 このエレメントは、DCS アプリケーションに関する問題判別に使用します。

host_prdid - ホスト製品/バージョン ID

サーバー上で実行中の製品とバージョン。

表 500. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl_info	基本

使用法 これを使用して、DRDA ホスト・データベース製品の製品とコード・バージョンを識別できます。 PPPVVRRM の形式になっています。各部分の定義は次のとおりです。

- PPP は、次のホスト DRDA 製品を示す。
  - ARI の場合 : DB2 Server for VSE & VM
  - DSN の場合 : DB2 for z/OS
  - QSQ の場合 : DB2 for i
  - SQL の場合 : 他の DB2 製品
- VV は 2 桁でバージョン番号を示す (バージョン番号が 1 桁の場合には、高位の桁は 0 になります)。
- RR は 2 桁でリリース番号を示す (リリース番号が 1 桁の場合には高位の桁は 0 になります)。
- M は 1 文字で修正レベルを示します (0-9 または A-Z)。

host_response_time ホスト応答時間

DCS ステートメント・レベルでは、ステートメントが DB2 Connect ゲートウェイから処理のためにホストに送信された時刻と、ホストから結果を受信した時刻との間の経過時間です。 DCS データベースおよび DCS アプリケーションのレベルでは、特定のアプリケーションまたはデータベースに対して実行されたすべてのステートメントについての経過時間の合計です。データ伝送レベルでは、この多数のデータ伝送を使用したすべてのステートメントに関するホスト応答時間の合計です。

表 501. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント、タイム・スタンプ
DCS ステートメント	dcs_stmt	ステートメント、タイム・スタンプ
データ伝送	stmt_transmissions	ステートメント、タイム・スタンプ

ステートメント・レベルでのスナップショット・モニターの場合、このカウンターはリセットできません。その他のレベルではこのカウンターはリセットできます。

使用法 このエレメントと送信されたアウトバウンド・バイト数および受信されたアウトバウンド・バイト数を組み合わせて使用すると、次のようにしてアウトバウンド応答時間 (転送速度) を計算できます。

(送信されたアウトバウンド・バイト数 + 受信されたアウトバウンド・バイト数) / ホストの応答時間

idle_agents - アイドル・エージェント数 :

アプリケーションにまだ割り当てられておらず、『アイドル』状態でエージェント・プール内に存在するエージェントの数。

表 502. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法 このエレメントは、 *num_poolagents* 構成パラメーターを設定するときに利用できます。アイドル・エージェントを持つことでエージェントの要求を処理できるので、パフォーマンスが向上します。

iid - 索引 ID : モニター・エレメント

索引の ID。

表 503. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

inbound_bytes_received 受信されたインバウンド・バイト数

DB2 Connect ゲートウェイがクライアントから受信したバイト数。ただし、通信プロトコルのオーバーヘッド (例えば TCP/IP や SNA のヘッダー) は含まれません。

表 504. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl	基本
DCS ステートメント	dcs_stmt	ステートメント

アプリケーション・レベルでのスナップショット・モニターの場合、このカウンターはリセットできます。その他のレベルではこのカウンターはリセットできません。

使用法 このエレメントを使用して、クライアントから DB2 Connect ゲートウェイへのスループットを測定します。

inbound_bytes_sent 送信されたインバウンド・バイト数

DB2 Connect ゲートウェイがクライアントに送信したバイト数。通信プロトコルのオーバーヘッド (例えば TCP/IP や SNA のヘッダー) は含まれません。

表 505. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl	基本
DCS ステートメント	dcs_stmt	ステートメント

アプリケーション・レベルでのスナップショット・モニターの場合、このカウンターはリセットできます。その他のレベルではこのカウンターはリセットできません。

使用法 このエレメントを使用して、 DB2 Connect ゲートウェイからクライアントへのスループットを測定します。

inbound_comm_address インバウンド通信アドレス

これはクライアントの通信アドレスです。例えば、SNA ネット ID と LU パートナー名、または TCP/IP 用の IP アドレスとポート番号などです。

表 506. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
DCS アプリケーション	dcs_appl_info	基本

使用法 このエレメントは、DCS アプリケーションに関する問題判別に使用します。

include_col_updates - 列の更新の組み込み : モニター・エレメント

列の更新の組み込みの数。

表 507. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

index_object_pages 索引オブジェクト・ページ数

表に対して定義されたすべての索引が使用するディスク・ページの数。

表 508. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本

表 509. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表	event_table	-

使用法 このエレメントを使用すると、特定の表に対して定義された索引が使用する実際のスペースの量を表示できます。このエレメントと表イベント・モニターを組み合わせて使用すると、時間とともに索引が大きくなる比率を追跡できます。このエレメントは、パーティション表では戻されません。

index_only_scans - 索引のみのスキャン : モニター・エレメント

索引のみのスキャンの数。索引のみのスキャンは、スキャンの結果が索引へのアクセスのみで得られた場合に生じます。

表 510. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

index_scans - 索引スキャン : モニター・エレメント

索引スキャンの数。

表 511. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

index_tbsp_id - 索引表スペース ID : モニター・エレメント

この表で作成された索引を保持する表スペースの ID。

表 512. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLE 表関数 - 表メトリックの取得	常に収集される

使用法

このエレメントの値は、SYSCAT.TABLESPACES のビューの TBSPACEID 列の値と一致します。

input_db_alias 入力データベース別名

スナップショット関数を呼び出すときに指定するデータベースの別名。

表 513. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl_id_info	基本
表スペース	tablespace_list	バッファー・プール
バッファー・プール	バッファー・プール	バッファー・プール
表	table_list	表
ロック	db_lock_list	基本

使用法 このエレメントを使用すると、モニター・データが適用される特定のデータベースを識別できます。特定のデータベースに関連するモニター情報を要求したとき以外は、このエレメントはブランクとなります。

1 つのデータベースが複数の別名を持つことがあるので、このフィールドの値と *client_db_alias* モニター・エレメントの値とが異なる場合があります。複数のアプリケーションやユーザーが異なる別名を使用して、同じデータベースに接続することができます。

insert_sql_stmts 挿入回数

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時か、またはデータベース・モニター・カウンターの最後のリセット時のどちらか最近の方の時点以降に、いずれかのアプリケーションに代わってフェデレーテッド・サーバーがこのデータ・ソースに INSERT ステートメントを発行した合計回数が含まれています。

表 514. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

このエレメントを使用すると、フェデレーテッド・サーバーまたはアプリケーションによりこのデータ・ソースに対して行われたデータベース・アクティビティーのレベルを判別できます。

このエレメントを使用すると、次の公式を使用して、フェデレーテッド・サーバーまたはアプリケーションによるこのデータ・ソースへの書き込みアクティビティーのパーセンテージも判別できます。

```
書き込みアクティビティー =
   (INSERT ステートメント + UPDATE ステートメント +
   DELETE ステートメント) /
   (SELECT ステートメント + INSERT ステートメント +
   UPDATE ステートメント +
   DELETE ステートメント)
```

insert_time 挿入応答時間

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時か、またはデータベース・モニター・カウンターの最後のリセット時のどちらか最近の方の時点以降に、このフェデレーテッド・サーバー・インスタンス上で実行されているすべてのアプリケーションまたは単一アプリケーションからの INSERT に対して、このデータ・ソースが応答に要した合計時間が含まれています (ミリ秒単位)。

応答時間とは、フェデレーテッド・サーバーが INSERT ステートメントをデータ・ソースにサブミットしてからデータ・ソースが INSERT を処理したことをフェデレーテッド・サーバーに応答するまでの時間です。

表 515. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	タイム・スタンプ
アプリケーション	appl_remote	タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

このエレメントを使用すると、このデータ・ソースに対する INSERT が処理されるのを待機するために生じた実際の時間を判別できます。この情報は、キャパシティー・プランニングおよびチューニングを行うときに便利です。

insert_timestamp - ステートメント挿入タイムスタンプ : モニター・エレメント

ステートメントがキャッシュに入れられた時刻。

表 516. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 517. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
動的 SQL	dynsql	基本

使用法

このエレメントは、ステートメントがキャッシュに入れられた時刻を指定します。これを使用して、キャッシュ内でのステートメントの存続時間を見積もることができます。

int_auto_rebinds 内部自動再バインド

試行された自動再バインド (または再コンパイル) の数。

表 518. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 519. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 自動再バインドは、パッケージが無効にされている場合にシステムが実行する内部バインドです。再バインドは、データベース・マネージャーが初めてパッケージから SQL ステートメントを実行する必要が生じたときに行われます。パッケージは、例えば次の場合に無効にされます。

- プランが従属している表、ビュー、または索引などのオブジェクトのドロップ。
- 外部キーの追加またはドロップ。
- プランが従属しているオブジェクト特権の取り消し。

このエレメントを使用すると、アプリケーション・レベルまたはデータベース・レベルのデータベース・アクティビティーのレベルを判別できます。int_auto_rebinds はパフォーマンスに大きな影響を与えるので、できるだけ最小限に抑える必要があります。

このエレメントを使用すると、次の公式を使用して、再バインド・アクティビティーのパーセンテージも計算できます。

```
int_auto_rebinds / total number of statements
```

この情報は、アプリケーションのアクティビティーおよびスループットの分析に役立ちます。

int_commits 内部コミット数

データベース・マネージャーによって内部で開始されたコミットの合計数。

表 520. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 521. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 内部コミットは、以下のいずれかの間に発生する場合があります。

- 再編成
- インポート
- バインドまたはプリコンパイル
- 明示的な SQL COMMIT ステートメントを実行せずにアプリケーションが終了した場合 (UNIX の場合)

この値には明示的な SQL COMMIT ステートメントは含まれず、次の時点以降のこれらの内部コミットの数を表します。

- データベースへの接続 (データベース・レベルの情報の場合は、最初の接続時刻)
- データベース・モニター・カウンターの最後のリセット

このエレメントを使用すると、次の項目を合計して合計作業単位数を計算できます。

```
  commit_sql_stmts
+ int_commits
+ rollback_sql_stmts
+ int_rollbacks
```

注: 計算した作業単位に含まれるのは、次の時点以降の作業単位だけです。

- データベースへの接続 (データベース・レベルの情報の場合は、最初の接続時刻)
- データベース・モニター・カウンターの最後のリセット

この計算は、アプリケーションまたはデータベース・レベルで実行できます。

int_deadlock_rollbacks デッドロックによる内部ロールバック

デッドロックのためにデータベース・マネージャーによって開始された強制ロールバックの合計数。ロールバックは、デッドロックを解決するためにデータベース・マネージャーが選択したアプリケーション内の現在の作業単位上で実行されます。

エレメント ID

int_deadlock_rollbacks

エレメント・タイプ

カウンター

表 522. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 523. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	-

使用法 このエレメントは中断されたデッドロックの数を示しており、並行性の問題の標識として使用できます。 int_deadlock_rollbacks は、データベースのスループットが低下するため、この問題は重要です。

この値は、int_rollbacks が示す値に含まれています。

int_node_splits - 中間ノードの分割 : モニター・エレメント

挿入操作時に中間索引ノードが分割された回数。

表 524. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

int_rollbacks 内部ロールバック数

データベース・マネージャーによって内部で開始されたロールバックの合計数。

エレメント ID

int_rollbacks

エレメント・タイプ

カウンター

表 525. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 526. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 内部ロールバックは、以下のいずれかが正常に完了**できない**ときに起こります。

- 再編成
- インポート
- バインドまたはプリコンパイル
- デッドロック状態またはロック・タイムアウト状態によりアプリケーションが終了した場合
- 明示的なコミットまたはロールバック・ステートメントを実行せずにアプリケーションが終了した場合 (Windows の場合)

この値は、次の時点以降のこれらの内部ロールバックの数を表します。

- データベースへの接続 (データベース・レベルの情報の場合は、最初の接続時刻)
- データベース・モニター・カウンターの最後のリセット

この値には明示的な SQL ROLLBACK ステートメントは含まれませんが、int_deadlock_rollbacks のカウントは含まれます。

このエレメントを使用すると、次の項目を合計して合計作業単位数を計算できます。

```
   commit_sql_stmts
 + int_commits
 + rollback_sql_stmts
 + int_rollbacks
```

注: 計算した作業単位には、次の時点以降の作業単位が含まれます。

- データベースへの接続 (データベース・レベルの情報の場合は、最初の接続時刻)
- データベース・モニター・カウンターの最後のリセット

この計算は、アプリケーションまたはデータベース・レベルで実行できます。

int_rows_deleted 削除された内部行数

これは、内部アクティビティーの結果としてデータベースから削除された行の数です。

表 527. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本
アプリケーション	stmt	基本
動的 SQL	dynsql	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 528. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-
ステートメント	event_stmt	-

使用法 このエレメントを使用すると、ユーザーが気付かないようなデータベース・マネージャー内の内部アクティビティーを把握できます。このアクティビティーが高い場合は、表の設計内容を検討して、データベースに定義した参照制約やトリガーが必要かどうかを判別してください。

内部の削除アクティビティーは、次の原因により起こります。

- カスケード削除により ON CASCADE DELETE 参照制約が強制された場合。
- トリガーが起動された場合。

int_rows_inserted 挿入された内部行数

トリガーによって行われた内部アクティビティーの結果としてデータベースに挿入された行の数。

表 529. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本
アプリケーション	stmt	基本
動的 SQL	dynsql	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 530. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

表 530. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	-
ステートメント	event_stmt	-

使用法 このエレメントを使用すると、データベース・マネージャー内の内部アクティビティーを把握できます。このアクティビティーが高い場合は、このアクティビティーを低減するように設計を変更できるかどうか、検討してください。

int_rows_updated 更新された内部行数

これは、内部アクティビティーの結果としてデータベースから更新された行の数です。

表 531. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本
アプリケーション	stmt	基本
動的 SQL	dynsql	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 532. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-
ステートメント	event_stmt	-

使用法 このエレメントを使用すると、ユーザーが気付かないようなデータベース・マネージャー内の内部アクティビティーを把握できます。このアクティビティーが高い場合は、表の設計内容を検討して、データベースに定義した参照制約が必要かどうかを判別してください。

内部の更新アクティビティーは、次の原因により起こります。

- *set null* 行更新により ON DELETE SET NULL ルールに定義した参照制約が強制された場合。
- トリガーが起動された場合。

invocation_id - 呼び出し ID : モニター・エレメント

このアクティビティーの 1 つの特定の呼び出しを、同じネスト・レベルの他の呼び出しと区別する ID。

表 533. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

ipc_recv_volume - プロセス間通信の受信ボリューム : モニター・エレメント

データ・サーバーがクライアントから IPC 経由で受信したデータの量。この値はバイト単位で示されます。

表 534. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 535. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

ipc_recv_wait_time - プロセス間通信受信待機時間 : モニター・エレメント

IPC 通信プロトコルを使用して着信クライアント要求を受信するのにエージェントが費やした時間。値はミリ秒単位で報告されます。

表 536. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 537. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

ipc_recvs_total - プロセス間通信受信の合計 : モニター・エレメント

データベース・サーバーがクライアント・アプリケーションから IPC を使用してデータを受信した回数。

表 538. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 539. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

ipc_send_volume - プロセス間通信の送信ボリューム : モニター・エレメント

データ・サーバーによってクライアントに IPC プロトコル経由で送信されたデータの量。この値はバイト単位で示されます。

表 540. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 540. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 541. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

ipc_send_wait_time - プロセス間通信送信待機時間 : モニター・エレメント

クライアントへの IPC 送信のブロックにかかった時間。値はミリ秒単位で示されます。

表 542. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE

表 542. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 543. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

ipc_sends_total - プロセス間通信送信の合計 : モニター・エレメント

データベース・サーバーによってクライアント・アプリケーションに IPC を使用してデータが送信された回数。

表 544. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 544. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 545. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

is_system_appl システム・アプリケーション : モニター・エレメント

アプリケーションがシステム・アプリケーションであるかどうかを示します。

表 546. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本

使用法

is_system_appl モニター・エレメントは、アプリケーションが内部システム・アプリケーションであるかどうかを示します。可能な値は以下のとおりです。

0 ユーザー・アプリケーション

1 システム・アプリケーション

key_updates - キーの更新 : モニター・エレメント

キーの更新の数。

表 547. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

last_active_log 最終アクティブ・ログ・ファイル番号

最後のアクティブ・ログ・ファイルのファイル番号。

エレメント ID

last_active_log

エレメント・タイプ

情報

表 548. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	detail_log	基本

表 549. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 このエレメントと *first_active_log* および *current_active_log* エレメントを組み合わせて使用すると、アクティブ・ログ・ファイルの範囲を判別できます。アクティブ・ログ・ファイルの範囲を知っていると、ログ・ファイルに必要なディスク・スペースを判別するのに役立ちます。

また、このエレメントを使用すると、どのログ・ファイルにデータがあるか判別でき、スプリット・ミラー・サポートが必要なログ・ファイルを識別するのに役立ちます。

last_backup 最終バックアップ・タイム・スタンプ

データベース・バックアップが最後に完了した日時。

エレメント ID

last_backup

エレメント・タイプ

timestamp

表 550. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	タイム・スタンプ

使用法 このエレメントを使用すると、バックアップをしばらく行っていないデータベースを識別したり、最新のデータベース・バックアップ・ファイルを識別できます。データベースを一度もバックアップしていない場合は、このタイム・スタンプがゼロに初期化されます。

last_extent - 移動した最終エクステント : モニター・エレメント

表スペース・リバランサー・プロセスにより移動された最終エクステントの数値 ID。

表 551. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_EXTENT_MOVEMENT_STATUS - エクステントの移動の進行状況メトリックの取得	常に収集される

last_overflow_time 最後のイベント・オーバーフロー時刻

このオーバーフロー・レコードに記録されている最後のオーバーフローの日時。

表 552. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
オーバーフロー・レコード	event_overflow	-

使用法 このエレメントと *first_overflow_time* を組み合わせて使用すると、オーバーフロー・レコードを生成するのに要した経過時間を計算できます。

last_reference_time - 最終参照時刻 : モニター・エレメント

アクティビティーが要求によって最後にアクセスされた時刻。

表 553. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

last_reset 最後のリセット・タイム・スタンプ

GET SNAPSHOT を発行するアプリケーションのモニター・カウンターがリセットされた日時を示します。

エレメント ID

last_reset

エレメント・タイプ

timestamp

表 554. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	タイム・スタンプ
データベース	dbase	タイム・スタンプ
アプリケーション	appl	タイム・スタンプ
表スペース	tablespace_list	バッファー・プール、タイム・スタンプ
表	table_list	タイム・スタンプ

表 554. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	タイム・スタンプ
DCS アプリケーション	dcs_appl	タイム・スタンプ

使用法 このエレメントを使用すると、データベース・システム・モニターによって戻された情報の有効範囲を判別できます。

カウンターがこれまでにリセットされたことがない場合は、このエレメントはゼロになります。

データベース・マネージャーのカウンターがリセットされるのは、ユーザーがすべてのアクティブ・データベースをリセットしたときだけです。

last_wlm_reset 最後にリセットされた時刻 : モニター・エレメント

このエレメントは、このタイプの統計イベント・レコードが最後に作成された時刻をローカル・タイム・スタンプの形式で示します。

表 555. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wlstats	-
統計	event_wcstats	-
統計	event_qstats	-

使用法

wlm_last_reset および **statistics_timestamp** モニター・エレメントを使用すると、イベント・モニター統計レコード中の統計が収集された期間を判別できます。収集間隔の開始時刻は **wlm_last_reset** で、終了時刻は **statistics_timestamp** です。

lob_object_pages LOB オブジェクト・ページ数

LOB データが使用するディスク・ページの数。

表 556. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本

表 557. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表	event_table	-

使用法 このエレメントを使用すると、特定の表中の LOB データが使用する実際のスペースの量を表示できます。このエレメントと表イベント・モニターを組み合わせて使用すると、時間とともに LOB データが大きくなる比率を追跡できます。

local_cons - ローカル接続 :

モニター中のデータベース・マネージャー・インスタンス内のデータベースに現在接続しているローカル・アプリケーションの数。

表 558. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

この数は、データベース・マネージャーで発生している並行処理のレベルを判別するのに役立ちます。この値は頻繁に変化するため、システム使用量の現実的な値を得るためには、長期にわたり特定のインターバルを設けてサンプルを採取する必要があります。

この数値に含まれるのは、データベース・マネージャーと同じインスタンスで開始したアプリケーションだけです。アプリケーションは接続されていますが、データベース内で作業単位を実行している場合としていない場合があります。

このエレメントと rem_cons_in モニター・エレメントを組み合わせて使用すると、**max_connections** 構成パラメーターの設定値を調整するときに利用できます。

local_cons_in_exec - データベース・マネージャーで実行中のローカル接続 :

モニター中のデータベース・マネージャー・インスタンス内のデータベースに現在接続していて、作業単位を現在処理しているローカル・アプリケーションの数。

表 559. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

この数は、データベース・マネージャーで発生している並行処理のレベルを判別するのに役立ちます。この値は頻繁に変化するため、システム使用量の現実的な値を得るためには、長期にわたり特定のインターバルを設けてサンプルを採取する必要があります。この数値に含まれるのは、データベース・マネージャーと同じインスタンスで開始したアプリケーションだけです。

このエレメントと **rem_cons_in_exec** モニター・エレメントを組み合わせて使用すると、**max_coordagents** 構成パラメーターの設定値を調整するときに利用できます。

以下の推奨事項は、非コンセントレーター構成のみに適用されます。コンセントレーターが使用可能になっている場合、DB2 は多数のクライアント接続を 1 つのもっと小さなコーディネーター・エージェントのプールに多重化します。この場合、普通は **rem_cons_in_exec** および **local_cons_in_exec** の合計が **max_coordagents** 値に近くなってもかまいません。

- **max_coordagents** を AUTOMATIC に設定する場合は、調整を加えないでください。
- **max_coordagents** を AUTOMATIC に設定せず、**max_coordagents** および **local_cons_in_exec** の合計が **max_coordagents** に近い同じ場合は、**max_coordagents** の値を増やしてください。

local_start_time - ローカル開始時刻 : モニター・エレメント

このアクティビティーがパーティション上で作業を開始した時刻。これはローカル時間です。アクティビティーがシステムに入ったものの、キューに入れられていてまだ実行を開始していない場合は、このフィールドは空ストリングになります。

表 560. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

lock_attributes ロック属性 : モニター・エレメント

ロックの属性。

表 561. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
ロック	lock	基本
ロック	lock_wait	基本

表 562. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
デッドロック[1]	lock	-
デッドロック[1]	event_dlconn	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

ロック属性の設定として次のものがあります。ロック属性の各設定は、sqlmon.h で定義されているビット・フラグ値に基づいています。

API 定数	説明
SQLM_LOCKATTR_WAIT_FOR_AVAIL	使用可能を待機
SQLM_LOCKATTR_ESCALATED	エスカレーションによる獲得
SQLM_LOCKATTR_RR_IN_BLOCK	ブロック内の RR ロック
SQLM_LOCKATTR_INSERT	ロックの挿入
SQLM_LOCKATTR_DELETE_IN_BLOCK	ブロック内の削除対象行
SQLM_LOCKATTR_RR	RR スキャンによるロック
SQLM_LOCKATTR_UPDATE_DELETE	行ロックの更新/削除
SQLM_LOCKATTR_ALLOW_NEW	新規ロック要求を許可
SQLM_LOCKATTR_NEW_REQUEST	新規ロック・リクエスター
SQLM_LOCKATTR_INDOUBT	未確定トランザクションが保持するロック。
SQLM_LOCKATTR_LOW_PRIORITY	優先順位の低いアプリケーションが保持するロック。

lock_count ロック・カウント : モニター・エレメント

保持されているロックに関するロックの数。

表 563. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
ロック	lock	基本

表 564. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
デッドロック[1]	lock	-
デッドロック[1]	event_dlconn	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

この値の範囲は 1 から 255 です。新規ロックが獲得されると増分し、ロックが解放されると減分されます。

lock_count モニター・エレメントの値が 255 のときは、トランザクション期間ロックが保持されていることを示します。この時点でロックが獲得または解放されても **lock_count** モニター・エレメントは増分または減分されなくなります。 **lock_count** モニター・エレメントは次の 2 つのいずれかの場合に値が 255 に設定されます。

1. 獲得されている新規ロックによって **lock_count** モニター・エレメント値が 255 回増分された場合。

2. トランザクション期間ロックが明示的に獲得された場合。例えば LOCK TABLE ステートメントや INSERT が使用された場合です。

lock_current_mode 変換前の元のロック・モード

ロック変換操作のとき、保持されているロックのタイプは変換前に完了になります。

表 565. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
ロック	lock	基本
ロック	lock_wait	基本

表 566. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
デッドロック[1]	lock	-
デッドロック[1]	event_dlconn	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

ロック変換のシナリオの例を挙げます。更新または削除操作のときにターゲット行の X ロックを待機するとします。トランザクションがその行で S または V ロックを保持している場合、変換が必要になります。この時点で、ロックが X ロックに変換されるのを待機している間は、**lock_current_mode** エレメントには S または V の値が割り当てられます。

lock_escalation ロック・エスカレーション : モニター・エレメント

ロック要求がロック・エスカレーションの一部として行われたかどうかを示します。

表 567. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
ロック	lock	ロック
ロック	lock_wait	ロック

表 568. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
デッドロック[1]	lock	-

表 568. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック[1]	event_dlconn	-
詳細付きデッドロック[1]	event_detailed_dlconn	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

このエレメントを使用すると、デッドロックの原因が分かりやすくなります。アプリケーションがロック・エスカレーションを起こすようなデッドロックが発生した場合は、ロック・メモリーの量を増やすか、または任意のアプリケーションが要求できるロックのパーセンテージを変更してください。

lock_escals - ロック・エスカレーション数 : モニター・エレメント

ロックが複数の行ロックから 1 つの表ロックにエスカレートされた回数。

表 569. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 569. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 570. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 571. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
接続	event_conn	-
トランザクション	event_xact	-

使用法 アプリケーションが保留するロックの合計数がそのアプリケーションで使用可能なロック・リスト・スペースの最大量に達した場合、またはすべてのアプリケーションが使用するロック・リスト・スペースが合計ロック・リスト・スペースに近くなると、ロックはエスカレートされます。使用可能なロック・リスト・スペースの量は、 **maxlocks** および **locklist** 構成パラメーターによって決まります。

1 つのアプリケーションが使用可能な最大ロック数に達して、エスカレートするロックがほかにない場合は、ほかのアプリケーションに割り振られているロック・リストのスペースが使用されます。ロック・リスト全体が満杯になるとエラーが起こります。

このデータ項目には、排他ロック・エスカレーションも含めて、すべてのロック・エスカレーションのカウントが含まれます。

過剰なロック・エスカレーションが起こる場合は、いくつかの原因が考えられます。

- 同時アプリケーションの数に対してロック・リスト・サイズ (**locklist**) が小さい場合。
- 各アプリケーションが使用できるロック・リストのパーセント値 (**maxlocks**) が小さい場合。

- 1 つ以上のアプリケーションが使用しているロックの数が多すぎる場合。

これらの問題を解決するには、次のようにしてください。

- **locklist** 構成パラメーター値を大きくする。
- **maxlocks** 構成パラメーター値を大きくする。
- 次の数式を使用して、ロック数の多いアプリケーション (**locks_held_top** モニター・エレメントを参照)、または大量のロック・リストを保持しているアプリケーションを識別する。

```
(((locks held * 36) / (locklist * 4096)) * 100)
```

ここで、値を **maxlocks** と比較します。これらのアプリケーションがロック・リストの多くを使用すると、ほかのアプリケーションでロック・エスカレーションを起こします。これらのアプリケーションでは行ロックではなく表ロックを使用する必要が生じる場合がありますが、表ロックが原因で **lock_waits** および **lock_wait_time** モニター・エレメントの値が増加することがあります。

lock_hold_count ロック保留カウント : モニター・エレメント

ロックに置かれている保留の数。保留は WITH HOLD 節に登録されたカーソルといくつかの DB2 ユーティリティーによってロック上に置かれます。保留があるロックはトランザクションがコミットされても保留解除されません。

表 572. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
ロック	lock	基本

表 573. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
デッドロック[1]	lock	-
デッドロック[1]	event_dlconn	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

lock_list_in_use 使用中のロック・リスト・メモリーの合計

使用中のロック・リスト・メモリーの合計量 (バイト単位)。

エレメント ID

lock_list_in_use

エレメント・タイプ

水準点

表 574. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 このエレメントと *locklist* 構成パラメーターを組み合わせて使用すると、ロック・リスト使用率を計算できます。ロック・リスト使用率が高い場合は、そのパラメーターのサイズを増やすことを考慮してください。

注: 使用率を計算する場合、 *locklist* 構成パラメーターが各 4K バイトのページ単位で割り振られるのに対し、モニター・エレメントの結果はバイト数で表されることに注意してください。

lock_mode - ロック・モード : モニター・エレメント

保持されているロックのタイプ。

表 575. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ロック
ロック	lock	ロック
ロック	lock_wait	ロック

表 576. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
デッドロック[1]	lock	-
デッドロック[1]	event_dlconn	-
詳細付きデッドロック[1]	event_detailed_dlconn	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

このモードは、リソースの競合の原因を判別するときに利用できます。

このエレメントは、調査するモニター情報のタイプにより、次の内容を示します。

- 1 つのアプリケーションがロック待ちをしているオブジェクトに対して、別のアプリケーションが保留しているロックのタイプ (アプリケーション・モニターおよびデッドロック・モニターのレベル)。
- このアプリケーションが保持しているオブジェクトのロック・タイプ (オブジェクト・ロック・レベル)。

このフィールドの値は以下のとおりです。

モード	ロックのタイプ	API 定数
	ロックなし	SQLM_LNON
IS	意図的共有ロック	SQLM_LOIS
IX	意図的排他ロック	SQLM_LOIX
S	共用ロック	SQLM_LOOS
SIX	意図的排他ロックで共有	SQLM_LSIX
X	排他ロック	SQLM_LOOX
IN	意図なし	SQLM_LOIN
Z	超排他ロック	SQLM_LOOZ
U	更新ロック	SQLM_LOOU
NS	スキャン共有ロック	SQLM_LONS
NW	次キー弱排他ロック	SQLM_LONW

lock_mode_requested 要求されているロック・モード : モニター・エレメント

アプリケーションが要求しているロック・モード。

表 577. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
ロック	lock_wait	ロック

表 578. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
デッドロック[1]	event_dlconn	-
詳細付きデッドロック[1]	event_detailed_dlconn	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

アプリケーションが要求したロックのモード。この値は、リソース競合の原因を判別するのに役立ちます。

lock_name ロック名 : モニター・エレメント

内部バイナリー・ロック名。このエレメントはロックのユニーク ID を示します。

表 579. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
ロック	lock	基本
ロック	lock_wait	lock_wait

表 580. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
デッドロック[1]	lock	-
デッドロック[1]	event_dlconn	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

lock_node ロック・ノード

ロックに関係しているノード。

表 581. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	サブセクション	ステートメント
デッドロック	event_dlconn	ステートメント
詳細付きデッドロック	event_detailed_dlconn	ステートメント

使用法 これはトラブルシューティングに使用できます。

lock_object_name ロック対象名

このエレメントは情報提供のみを目的としています。アプリケーションがロックを保留するオブジェクトの名前 (オブジェクト・ロック・レベルの情報)、またはアプリケーションがロックの取得を待機しているオブジェクトの名前 (アプリケーション・レベルおよびデッドロック・レベルの情報)。

注: このモニター・エレメントは、使用しないでください。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 582. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ロック
ロック	appl_lock_list	ロック
ロック	lock	基本

表 583. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	lock	-
デッドロック	event_dlconn	-
詳細付きデッドロック	event_detailed_dlconn	-

使用法 表レベルのロックの場合、SMS および DMS 表スペースのファイル ID (FID)。行レベルのロックの場合のオブジェクト名は行 ID (RID)。表スペース・ロックの場合のオブジェクト名はブランク。バッファー・プール・ロックの場合のオブジェクト名は、バッファー・プールの名前。

ロックを保留する表を判別するときは、ファイル ID は固有のものとは限らないため、ファイル ID ではなく、*table_name* および *table_schema* を使用します。

ロックを保留している表スペースを判別するときは、*tablespace_name* を使用します。

lock_object_type - 待機中のロック対象タイプ : モニター・エレメント

アプリケーションがロックを保持しているオブジェクトのタイプ (オブジェクト・ロック・レベルの情報)、またはアプリケーションがロックの取得を待機しているオブジェクトのタイプ (アプリケーション・レベルおよびデッドロック・レベルの情報)。

表 584. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ロック
ロック	appl_lock_list	ロック
ロック	lock	基本
ロック	lock_wait	ロック

表 585. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
デッドロック[1]	lock	-
デッドロック[1]	event_dlconn	-
詳細付きデッドロック[1]	event_detailed_dlconn	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

このエレメントは、リソースの競合の原因を判別するときに役立ちます。

オブジェクト・タイプ ID は sqlmon.h で定義されます。オブジェクトのタイプは、次のいずれかになります。

- 表スペース (sqlmon.h の SQLM_TABLESPACE_LOCK)
- 表
- バッファー・プール
- ブロック
- レコード (または行)
- データ・パーティション (sqlmon.h の SQLM_TABLE_PART_LOCK)
- 内部 (データベース・マネージャーが内部で保持するほかのタイプのロック)
- 自動サイズ変更
- 自動ストレージ。

lock_release_flags ロック保留解除フラグ : モニター・エレメント

ロック保留解除フラグ。

表 586. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
ロック	lock	基本
ロック	lock_wait	基本

表 587. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
デッドロック[1]	lock	-
デッドロック[1]	event_dlconn	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

保留解除フラグの設定として次のものがあります。各保留解除フラグは sqlmon.h で定義されているビット・フラグ値に基づいています。

API 定数	説明
SQLM_LOCKRELFLAGS_SQLCOMPILER	SQL コンパイラーによるロック
SQLM_LOCKRELFLAGS_UNTRACKED	非ユニーク、非追跡のロック

注: 割り当てられていないビットはすべてアプリケーション・カーソルに使用されます。

lock_status - ロック状況 : モニター・エレメント

ロックの内部状況を示します。

表 588. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
ロック	lock	基本

表 589. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
デッドロック[1]	lock	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

このエレメントは、アプリケーションがオブジェクトに対するロックを取得するために待機しているときに起きていることを明らかにするのに役立ちます。アプリケーションがすでに必要なオブジェクトのロックを取得しているように見える場合でも、同じオブジェクトについて異なるタイプのロックの取得を待たなければならないことがあります。

ロックの状況は、次のいずれかになります。

付与済み状態
アプリケーションは、**lock_mode** モニター・エレメントが指定する状態のロックを保有しています。

変換中状態
アプリケーションが保持ロックのタイプを変更しようとしています。例えば、共有ロックから排他ロックへの変更などです。

注: API ユーザーは、データベース・システム・モニターの定数の定義が含まれているヘッダー・ファイル sqlmon.h を参照してください。

lock_timeout_val ロック・タイムアウト値 : モニター・エレメント

アプリケーションが SET CURRENT LOCK TIMEOUT ステートメントを発行した時点のタイムアウト値 (秒単位) を示します。ステートメントが実行されていない場合は、データベース・レベルのロック・タイムアウトが示されます。

表 590. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本
アプリケーション	agent	基本

表 591. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-

使用法

SET CURRENT LOCK TIMEOUT ステートメントを使用して、アプリケーション・エージェントが表または索引のロックを待機する最大期間を指定できます。

アプリケーションがロックを待つ時間が長すぎる場合は、アプリケーション中で **lock_timeout_val** モニター・エレメント値を検査して、設定値が大きすぎるかどうか調べることができます。アプリケーション・ロジックにとって適切な場合は、アプリケーションに変更を加え、ロック・タイムアウト値を小さくして、アプリケーションをタイムアウトさせることができます。 SET CURRENT LOCK TIMEOUT ステートメントを使用して、この変更を行えます。

アプリケーションが頻繁にタイムアウトする場合は、ロック・タイムアウトの設定値が小さすぎるかどうかを検査し、該当する場合は大きくすることができます。

lock_timeouts - ロック・タイムアウト数 : モニター・エレメント

オブジェクトをロックするための要求が許可されずにタイムアウトになった回数。

表 592. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 592. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 593. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 594. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
接続	event_conn	-

使用法

このエレメントは、 **locktimeout** データベース構成パラメーターの設定値を調整するときに利用できます。通常の操作レベルと比較して、ロックのタイムアウト回数が多くなった場合は、ロックを長期にわたって保有しているアプリケーションがある可能性があります。その場合、このエレメントは、ロックおよびデッドロックに関する他のいくつかのモニター・エレメントを分析して、アプリケーションに問題があるかどうかを判別する必要があることを示している場合があります。

locktimeout データベース構成パラメーターの設定値が高すぎると、ロックのタイムアウト回数が極端に少なくなります。この場合は、アプリケーションがロックを取得するための待機時間が長くなります。

lock_wait_start_time ロック待機開始タイム・スタンプ

現在、別のアプリケーションによってロックされているオブジェクトに対するロックを取得するために、このアプリケーションが待機を開始した日時。

エレメント ID

lock_wait_start_time

エレメント・タイプ

timestamp

表 595. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ロック、タイム・スタンプ
ロック	lock_wait	ロック、タイム・スタンプ

表 596. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	event_dlconn	タイム・スタンプ
詳細付きデッドロック	event_detailed_dlconn	タイム・スタンプ

使用法 このエレメントは、リソース競合の重大度を判別するときに役立ちます。

lock_wait_time - ロック待機中の時間 : モニター・エレメント

ロック待機に費やされた合計経過時間。値はミリ秒単位で示されます。

表 597. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE

表 597. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 598. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	ロック
アプリケーション	appl	ロック
ロック	appl_lock_list	appl_lock_list

スナップショット・モニターの場合、このカウンターはリセットできます。

表 599. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
接続	event_conn	-
トランザクション	event_xact	-

使用法 データベース・レベルでは、このデータベース内ですべてのアプリケーションが 1 つのロックを待機した合計経過時間を示します。

アプリケーション接続およびトランザクションのレベルでは、この接続またはトランザクションがロックの付与を待機した合計経過時間を示します。

このエレメントの値に、現在もロック待機状態にあるエージェントのロック待機時間は含まれません。これにはすでにロック待機が完了したエージェントのロック待機時間のみが含まれます。

このエレメントと **lock_waits** モニター・エレメントを組み合わせて使用すると、平均ロック待機時間を計算できます。この計算は、データベース・レベルとアプリケーション接続レベルのいずれでも行えます。

経過時間を示すモニター・エレメントを使用するときは、次のことを考慮してください。

- 経過時間は、システム負荷の影響を受けるので、実行する処理数が多くなると、この経過時間の値は大きくなる。
- このエレメントをデータベース・レベルで計算する場合、データベース・システム・モニターはアプリケーション・レベルの時間を合計する。この

場合、同時に複数のアプリケーション処理が実行されていることがあるので、データベース・レベルでは時間が二重に計算されます。

意味のあるデータを提供するためには、上記の説明に従って平均ロック待機時間を計算してください。

lock_wait_time_top – ロック待機時間の最上位 : モニター・エレメント

ワークロードでの任意の要求のロック待機時間の最高水準点。単位はミリ秒です。lock_wait_time_top 最高水準点はワークロードに対して常に収集されます。要求メトリックが使用可能になっている場合にのみ、要求はこの最高水準点に寄与します。

表 600. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats	-

使用法

このエレメントを使用して、収集された時間間隔におけるワークロード用のパーティションでの要求の最大ロック待機時間を判別することができます。

lock_waits - ロック待機数 : モニター・エレメント

アプリケーションまたは接続がロックを待機した合計回数。

表 601. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 601. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 602. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 603. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
接続	event_conn	-

使用法 データベース・レベルでは、アプリケーションがデータベース内でロックを待機した合計回数を示します。

アプリケーション接続レベルでは、この接続がロックを要求し、ほかの接続がデータ上でロックを保留していたために待機した合計回数を示します。

このエレメントと **lock_wait_time** を組み合わせて使用すると、データベース・レベルの場合は平均ロック待機時間を計算できます。この計算は、データベース・レベルとアプリケーション接続レベルのいずれでも行えます。

平均ロック待機時間が長い場合は、多数のロックを保留するアプリケーションまたはロック・エスカレーションを起こしているアプリケーションを探します。これにより、必要に応じてアプリケーションをエスカレーションして並行性を改善します。エスカレーションが原因で平均ロック待機時間が長くなっている場合は、 **locklist** および **maxlocks** 構成パラメーターのどちらか、または両方の設定値が低すぎることが原因と考えられます。

locks_held ロック保持数

現在保持されているロックの数。

エレメント ID
locks_held

エレメント・タイプ
ゲージ

表 604. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本
ロック	db_lock_list	基本
ロック	appl_lock_list	基本

表 605. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
詳細付きデッドロック	event_detailed_dlconn	-

使用法 モニター情報がデータベース・レベルの場合は、データベース内のすべてのアプリケーションが現在保持しているロックの合計数を示します。

モニター情報がアプリケーション・レベルの場合は、アプリケーションのすべてのエージェントが現在保持しているロックの合計数を示します。

locks_held_top ロック保持最大数

このトランザクション中に保持されたロックの最大数。

エレメント ID
locks_held_top

エレメント・タイプ
カウンター

表 606. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
トランザクション	event_xact	-

使用法 このエレメントを使用すると、 *maxlocks* 構成パラメーターで定義されている、アプリケーションが使用可能な最大ロック数に近づいているかどうかを判別できます。このパラメーターは、ロック・エスカレーションを起こさずに各アプリケーションが使用できるロック・リストのパーセンテージを示します。ロック・エスカレーションが起こると、データベースに接続されている各アプリケーション間の並行性が低下します。

maxlocks パラメーターがパーセンテージで指定され、このエレメントはカウンターとなっているので、次の公式を使用すると、このエレメントが提供するカウントと 1 つのアプリケーションが保持できる合計ロック数を比較できます。

```
(locklist * 4096 / 36 ) * (maxlocks / 100)
```

ロック数が多い場合は、アプリケーション内で実行するコミットの数を多くして、一部のロックを解放する必要があります。

locks_in_list 報告されたロックの回数

イベント・モニターの報告対象の特定のアプリケーションが保留するロック数。

表 607. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
詳細付きデッドロック	event_detailed_dlconn	-

locks_waiting ロックで待機中の現行エージェント

ロック待機中のエージェントの数を示します。

表 608. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本
ロック	db_lock_list	基本

使用法 このエレメントと **appls_cur_cons** と組み合わせて使用すると、ロックを待機中のアプリケーションのパーセンテージが分かります。この値が大きい場合は、アプリケーションに並行性の問題がある可能性があるため、ロックや排他ロックを長時間にわたって保留しているアプリケーションを確認する必要があります。

log_buffer_wait_time - ログ・バッファー待機時間 : モニター・エレメント

ログ・バッファー内のスペースの待機にエージェントが費やした時間。値はミリ秒単位で示されます。

表 609. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 609. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 610. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

log_disk_wait_time - ログ・ディスク待機時間 : モニター・エレメント

ログ・レコードがディスクにフラッシュされるのをエージェントが待機していた時間。値はミリ秒単位で示されます。

表 611. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE

表 611. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 612. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

log_disk_waits_total - ログ・ディスク待機の合計 : モニター・エレメント

ログ・データがディスクに書き込まれるのをエージェントが待機しなければならなかった回数。

表 613. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 614. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

log_held_by_dirty_pages ダーティー・ページ別に計算されるログ・スペースの量

データベース中の最も古いダーティー・ページと、アクティブ・ログの先頭との間の差に対応するログの量 (バイト単位)。

エレメント ID

log_held_by_dirty_pages

エレメント・タイプ

水準点

表 615. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 616. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 この値は、スナップショットをとる際に、そのスナップショットの時点の条件に基づいて計算されます。

このエレメントは、バッファー・プール中の最も古いページに関するページ・クリーニングの有効性を評価するのに使用してください。

バッファー・プール中の古いページのクリーニングは、 *softmax* データベース構成パラメーターによって管理されます。ページ・クリーニングが有効な場合は、 *log_held_by_dirty_pages* がほぼ次の値以下である必要があります。

```
(softmax / 100)  *  logfilsiz * 4096
```

このステートメントが真でない場合は、ページ・クリーナー数 (*num_iocleaners*) 構成パラメーターを大きくしてください。

この条件が真で、ダーティー・ページに保持されるログを少なくするのが望ましい場合は、 *softmax* 構成パラメーターを小さくしてください。

log_read_time ログ読み取り時間

ロガーがログ・データをディスクから読み取るのに要した合計経過時間。

表 617. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 618. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 このエレメントと *log_reads*、 *num_log_read_io*、および *num_log_data_found_in_buffer* エレメントを組み合わせて使用すると、次のことを判別できます。

- 現行ディスクがロギングに適しているかどうか。
- ログ・バッファー・サイズが適切かどうか。

log_reads 読み取られたログ・ページの数

ロガーがディスクから読み取ったログ・ページの数。

エレメント ID

log_reads

エレメント・タイプ

カウンター

表 619. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 620. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 このエレメントとオペレーティング・システム・モニターを組み合わせて使用すると、データベース・アクティビティーによって発生した装置上の入出力の量がわかります。

log_to_redo_for_recovery リカバリーの場合に再実行されるログの量

クラッシュ・リカバリーの場合に再実行する必要のあるログの量 (バイト単位)。

エレメント ID

log_to_redo_for_recovery

エレメント・タイプ

水準点

表 621. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 622. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 この値は、スナップショットをとる際に、そのスナップショットの時点の条件に基づいて計算されます。値が大きいほど、システムのクラッシュ後のリカバリー時間が長くなることを示します。値が大きすぎるように思える場合は、 *log_held_by_dirty_pages* モニター・エレメントを検査して、ページ・クリーニングを調整する必要があるかどうか調べてください。また、終了する必要のある長期実行トランザクションがあるかどうかも検査してください。

log_write_time ログ書き込み時間

ロガーがログ・データをディスクに書き込むのに要した合計経過時間。

表 623. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 624. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 このエレメントと *log_writes* および *num_log_write_io* エレメントを組み合わせて使用すると、現行ディスクがロギングに適しているかどうかを判別できます。

log_writes 書き込まれたログ・ページの数

ロガーがディスクに書き込んだログ・ページの数。

エレメント ID
log_writes

エレメント・タイプ
カウンター

表 625. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 626. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 このエレメントとオペレーティング・システム・モニターを組み合わせて使用すると、データベース・アクティビティーによって発生した装置上の入出力の量がわかります。

注: ログ・ページがディスクに書き込まれる際、最後のページがいっぱいになっていない場合があります。その場合、部分的なログ・ページがログ・バッファー内に残り、さらにログ・レコードがページに書き込まれます。その結果、ログ・ページは、ロガーによってディスクに複数回書き込まれることがあります。このエレメントからは、 DB2 が生成したページ数は測定できません。

long_object_pages 長いオブジェクト・ページ数

表中の LONG データが使用するディスク・ページの数。

表 627. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本

表 628. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表	event_table	-

使用法 このエレメントを使用すると、特定の表中の LONG データが使用する実際のスペースの量を表示できます。このエレメントと表イベント・モニターを組み合わせて使用すると、時間とともに LONG データが大きくなる比率を追跡できます。

long_tbsp_id - LONG 表スペース ID : モニター・エレメント

この表で、LONG データ (LONG または LOB タイプの列) を保持する表スペースの ID。

表 629. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLE 表関数 - 表メトリックの取得	常に収集される

使用法

このエレメントの値は、SYSCAT.TABLESPACES のビューの TBSPACEID 列の値と一致します。

max_agent_overflows 最大エージェント・オーバーフロー回数 : モニター・エレメント

最大エージェント数 (**maxagents**) 構成パラメーターにすでに達しているときに、新規エージェント作成要求を受信した回数。

注: max_agent_overflows モニター・エレメントは、DB2 バージョン 9.5 以降では推奨されません。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 630. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

maxagents 構成パラメーターに達してもエージェント作成要求をまだ受け取る場合は、このノードのワークロードが高すぎることを示している可能性があります。

max_data_received_1024 受信アウトバウンド・バイト数が 513 から 1024 バイトのステートメント数

このエレメントは、受信アウトバウンド・バイト数が 513 から 1024 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID

max_data_received_1024

エレメント・タイプ

カウンター

表 631. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_received_128 受信アウトバウンド・バイト数が 1 から 128 バイトのステートメント数

このエレメントは、受信アウトバウンド・バイト数が 1 から 128 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID

max_data_received_128

エレメント・タイプ

カウンター

表 632. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_received_16384 受信アウトバウンド・バイト数が 8193 から 16384 バイトのステートメント数

このエレメントは、受信アウトバウンド・バイト数が 8193 から 16384 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID

max_data_received_16384

エレメント・タイプ

カウンター

表 633. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_received_2048 受信アウトバウンド・バイト数が 1025 から 2048 バイトのステートメント数

このエレメントは、受信アウトバウンド・バイト数が 1025 から 2048 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID

max_data_received_2048

エレメント・タイプ

カウンター

表 634. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_received_256 受信アウトバウンド・バイト数が 129 から 256 バイトのステートメント数

このエレメントは、受信アウトバウンド・バイト数が 129 から 256 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID

max_data_received_256

エレメント・タイプ

カウンター

表 635. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_received_31999 受信アウトバウンド・バイト数が 16385 から 31999 バイトのステートメント数 : モニター・エレメント

このエレメントは、受信アウトバウンド・バイト数が 16385 から 31999 バイトだったステートメントまたはチェーンの数を示します。

表 636. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_received_4096 受信アウトバウンド・バイト数が 2049 から 4096 バイトのステートメント数

このエレメントは、受信アウトバウンド・バイト数が 2049 から 4096 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID
max_data_received_4096

エレメント・タイプ
カウンター

表 637. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_received_512 受信アウトバウンド・バイト数が 257 から 512 バイトのステートメント数

このエレメントは、受信アウトバウンド・バイト数が 257 から 512 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID
max_data_received_512

エレメント・タイプ
カウンター

表 638. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_received_64000 受信アウトバウンド・バイト数が 32000 から 64000 バイトのステートメント数 : モニター・エレメント

このエレメントは、受信アウトバウンド・バイト数が 32000 から 64000 バイトだったステートメントまたはチェーンの数を示します。

表 639. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_received_8192 受信アウトバウンド・バイト数が 4097 から 8192 バイトのステートメント数

このエレメントは、受信アウトバウンド・バイト数が 4097 から 8192 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID

max_data_received_8192

エレメント・タイプ

カウンター

表 640. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_received_gt64000 受信アウトバウンド・バイト数が 64000 バイトを超えるステートメント数

このエレメントは、受信アウトバウンド・バイト数が 64000 バイトを超えたステートメントまたはチェーンの数を示します。

エレメント ID

max_data_received_gt64000

エレメント・タイプ
カウンター

表 641. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_1024 送信アウトバウンド・バイト数が 513 から 1024 バイトのステートメント数

このエレメントは、送信アウトバウンド・バイト数が 513 から 1024 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID
max_data_sent_1024

エレメント・タイプ
カウンター

表 642. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_128 送信アウトバウンド・バイト数が 1 から 128 バイトのステートメント数

このエレメントは、送信アウトバウンド・バイト数が 1 から 128 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID
max_data_sent_128

エレメント・タイプ
カウンター

表 643. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_16384 送信アウトバウンド・バイト数が 8193 から 16384 バイトのステートメント数

このエレメントは、送信アウトバウンド・バイト数が 8193 から 16384 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID

max_data_sent_16384

エレメント・タイプ

カウンター

表 644. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_2048 送信アウトバウンド・バイト数が 1025 から 2048 バイトのステートメント数

このエレメントは、送信アウトバウンド・バイト数が 1025 から 2048 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID

max_data_sent_2048

エレメント・タイプ

カウンター

表 645. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_256 送信アウトバウンド・バイト数が 129 から 256 バイトのステートメント数

このエレメントは、送信アウトバウンド・バイト数が 129 から 256 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID

max_data_sent_256

エレメント・タイプ

カウンター

表 646. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_31999 送信アウトバウンド・バイト数が 16385 から 31999 バイトのステートメント数

このエレメントは、送信アウトバウンド・バイト数が 16385 から 31999 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID

max_data_sent_31999

エレメント・タイプ

カウンター

表 647. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_4096 送信アウトバウンド・バイト数が 2049 から 4096 バイトのステートメント数

このエレメントは、送信アウトバウンド・バイト数が 2049 から 4096 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID

max_data_sent_4096

エレメント・タイプ

カウンター

表 648. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_512 送信アウトバウンド・バイト数が 257 から 512 バイトのステートメント数

このエレメントは、送信アウトバウンド・バイト数が 257 から 512 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID

max_data_sent_512

エレメント・タイプ

カウンター

表 649. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_64000 送信アウトバウンド・バイト数が 32000 から 64000 バイトのステートメント数

このエレメントは、送信アウトバウンド・バイト数が 32000 から 64000 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID

max_data_sent_64000

エレメント・タイプ

カウンター

表 650. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_8192 送信アウトバウンド・バイト数が 4097 から 8192 バイトのステートメント数

このエレメントは、送信アウトバウンド・バイト数が 4097 から 8192 バイトだったステートメントまたはチェーンの数を示します。

エレメント ID

max_data_sent_8192

エレメント・タイプ

カウンター

表 651. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_data_sent_gt64000 送信アウトバウンド・バイト数が 64000 バイトを超えるステートメント数

このエレメントは、送信アウトバウンド・バイト数が 64000 バイトを超えたステートメントまたはチェーンの数を示します。

エレメント ID

max_data_sent_gt64000

エレメント・タイプ

カウンター

表 652. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_network_time_100_ms ネットワーク時間が 16 から 100 ミリ秒のステートメント数

このエレメントは、ネットワーク時間が 16 ミリ秒を超えて 100 ミリ秒以下だったステートメントまたはチェーンの数を示します (ネットワーク時間は、 1 つのステートメントまたはチェーンに対するホスト応答時間と実行経過時間の差です)。

表 653. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_network_time_16_ms ネットワーク時間が 4 から 16 ミリ秒のステートメント数

このエレメントは、ネットワーク時間が 4 ミリ秒を超えて 16 ミリ秒以下だったステートメントまたはチェーンの数を示します (ネットワーク時間は、 1 つのステートメントまたはチェーンに対するホスト応答時間と実行経過時間の差です)。

エレメント ID

max_network_time_16_ms

エレメント・タイプ

カウンター

表 654. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_network_time_1_ms ネットワーク時間が 1 ミリ秒以下のステートメント数

このエレメントは、ネットワーク時間が 1 ミリ秒以下だったステートメントまたはチェーンの数を示します (ネットワーク時間は、 1 つのステートメントまたはチェーンに対するホスト応答時間と実行経過時間の差です)。

表 655. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_network_time_4_ms ネットワーク時間が 1 から 4 ミリ秒のステートメント数

このエレメントは、ネットワーク時間が 1 ミリ秒を超えて 4 ミリ秒以下だったステートメントまたはチェーンの数を示します (ネットワーク時間は、 1 つのステートメントまたはチェーンに対するホスト応答時間と実行経過時間の差です)。

エレメント ID

max_network_time_4_ms

エレメント・タイプ

カウンター

表 656. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_network_time_500_ms ネットワーク時間が 100 から 500 ミリ秒のステートメント数

このエレメントは、ネットワーク時間が 100 ミリ秒を超えて 500 ミリ秒以下だったステートメントまたはチェーンの数を示します (ネットワーク時間は、 1 つのステートメントまたはチェーンに対するホスト応答時間と実行経過時間の差です)。

表 657. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

max_network_time_gt500_ms ネットワーク時間が 500 ミリ秒を超えるステートメント数

このエレメントは、ネットワーク時間が 500 ミリ秒を超えたステートメントまたはチェーンの数を示します (ネットワーク時間は、 1 つのステートメントまたはチェーンに対するホスト応答時間と実行経過時間の差です)。

表 658. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント
DCS アプリケーション	dcs_appl	ステートメント
データ伝送	stmt_transmissions	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

member - データベース・メンバー・モニター・エレメント

この結果レコードのデータの取得元となったデータベース・メンバーの数値 ID。

表 659. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得	常に収集される
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得	常に収集される
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	常に収集される
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	常に収集される
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	常に収集される
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	常に収集される

表 659. 表関数モニター情報 (続き)

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	常に収集される
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	常に収集される
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得	常に収集される
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	常に収集される
MON_GET_EXTENT_MOVEMENT_STATUS - エクステントの移動の進行状況メトリックの取得	常に収集される
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

表 660. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

DB2 メンバーは、単一ホスト上で DB2 サーバー・ソフトウェアを実行するデータベース・マネージャー・インスタンスであり、DB2 メンバーはそれに接続するアプリケーションからのデータベース要求を受け入れて処理します。

message コントロール表メッセージ

MESSAGE_TIME 列内のタイム・スタンプの性質。このエレメントは、表書き込みイベント・モニターによってコントロール表でのみ使用されます。

表 661. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
-	-	-

使用法

次の値が使用されます。

DROPPED RECORDS: *n*
MONHEAP を割り振れなかったためにドロップされたアクティビティー・レコードの数。

FIRST_CONNECT
活動化後のデータベースへの最初の接続の時刻。

EVMON_START
EVMONNAME 列にリストされているイベント・モニターが開始された時刻。

OVERFLOWS: *n*
バッファー・オーバーフローのために *n* 個のレコードが廃棄されたことを示します。

LAST DROPPED RECORD
アクティビティー・レコードが最後にドロップされた時刻。

message_time タイム・スタンプ・コントロール表メッセージ

MESSAGE 列に記述されているイベントに対応したタイム・スタンプ。このエレメントは、表書き込みイベント・モニターによってコントロール表でのみ使用されます。

表 662. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
-	-	-

nesting_level - ネスト・レベル : モニター・エレメント

これは、このアクティビティーのネスト・レベルを表します。ネスト・レベルは、最上位の親アクティビティー内でこのアクティビティーがネストされている深さのことです。

表 663. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

network_time_bottom ステートメントの最小ネットワーク時間

このエレメントは、 DCS データベースに対してまたはこの DCS アプリケーション内で実行されたステートメントについて、あるいはこの回数のデータ伝送を使用したステートメントについて、最小ネットワーク時間を示します (ネットワーク時間は、1 つのステートメントに対するホスト応答時間と実行経過時間の差です)。

エレメント ID
network_time_bottom

エレメント・タイプ
水準点

表 664. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント、タイム・スタンプ
DCS アプリケーション	dcs_appl	ステートメント、タイム・スタンプ
データ伝送	stmt_transmissions	ステートメント、タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。

network_time_top ステートメントの最大ネットワーク時間

このエレメントは、 DCS データベースに対してまたはこの DCS アプリケーション内で実行されたステートメントについて、あるいはこの回数のデータ伝送を使用したステートメントについて、最長ネットワーク時間を示します (ネットワーク時間は、1 つのステートメントに対するホスト応答時間と実行経過時間の差です)。

エレメント ID
network_time_top

エレメント・タイプ
水準点

表 665. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	ステートメント、タイム・スタンプ
DCS アプリケーション	dcs_appl	ステートメント、タイム・スタンプ
データ伝送	stmt_transmissions	ステートメント、タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データベース・レベルおよびアプリケーション・レベルでのデータベース・アクティビティーやネットワーク・トラフィックの状態がより明確になります。「タイム・スタンプ」スイッチが OFF のときは、このエレメントは収集されません。

nleaf - リーフ・ページ数 : モニター・エレメント

リーフ・ページのおおよその数。

表 666. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

nlevels - 索引レベル数 : モニター・エレメント

索引レベルの数。これは近似値です。

表 667. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

node_number ノード番号

db2nodes.cfg ファイル内でノードに割り当てられた番号。

表 668. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	collected	基本
データベース・マネージャー	memory_pool	基本
データベース・マネージャー	fcm	基本
データベース・マネージャー	fcm_node	基本
データベース・マネージャー	utility_info	基本
データベース	detail_log	基本
バッファー・プール	bufferpool_nodeinfo	バッファー・プール
表スペース	rollforward	基本
ロック	lock	基本
ロック	lock_wait	基本
データベース	db_sto_path_info	バッファー・プール

表 669. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_connheader	-
デッドロック	lock	-
オーバーフロー・レコード	event_overflow	-
データベース	event_dbmemuse	-
接続	event_connmemuse	-

使用法 この値は現在のノード番号を示しており、複数のノードをモニターするときにこの値を利用できます。

nonboundary_leaf_node_splits - 非境界リーフ・ノードの分割 : モニター・エレメント

挿入操作時に非境界リーフ・ノードが分割された回数。

表 670. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

num_agents ステートメントで作動しているエージェントの数

現在ステートメントまたはサブセクションを実行している並行エージェントの数。

エレメント ID

num_agents

エレメント・タイプ

ゲージ

表 671. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント
アプリケーション	サブセクション	ステートメント

使用法 照会の並列処理の度合いを示します。スナップショットを連続的にとることによって、照会の実行進行状況を追跡するときに役に立ちます。

num_assoc_agents 関連したエージェント数

これは、アプリケーション・レベルでは、 1 つのアプリケーションに関連付けられているサブエージェントの数です。データベース・レベルでは、すべてのアプリケーション用のサブエージェントの数です。

エレメント ID

num_assoc_agents

エレメント・タイプ

ゲージ

表 672. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl_info	基本

使用法 このエレメントは、エージェント構成パラメーターの設定を評価するのに役立ちます。

num_compilations ステートメント・コンパイル数

特定の SQL ステートメントに対する異なるコンパイルの数。

表 673. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
動的 SQL	dynsql	基本

使用法 異なるスキーマで発行された SQL ステートメント (″select t1 from foo″ など) の中には、それらが異なるアクセス・プランを参照する場合でも DB2 キャッシュ内では同じステートメントと見なされるものがあります。この値と num_executions を組み合わせて使用すると、コンパイル環境に問題があるために動的 SQL スナップショット統計の結果に狂いが生じていないかどうかを判別できます。

num_db_storage_paths 自動ストレージ・パスの数

このエレメントは、このデータベースに関連した自動ストレージ・パスの数を示します。

表 674. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 このエレメントを db_storage_path モニター・エレメントと一緒に使用して、このデータベースに関連したストレージ・パスを識別できます。

num_executions - ステートメント実行回数 : モニター・エレメント

SQL ステートメントが実行された回数。

表 675. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 676. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
動的 SQL	dynsql	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

このエレメントを使用して、システムで最も頻繁に実行された SQL ステートメントを識別できます。

ステートメントごとに報告されたアクティビティー・メトリックについて、パッケージ・キャッシュ・レベルでの平均を計算するには、このエレメントを使用します。例えば、パッケージ・キャッシュ・レベルで報告されたあるステートメントの実行当たりの平均 CPU 使用量は、次の数式で計算できます。

```
total_cpu_time / num_exec_with_metrics
```

平均を計算する場合は、**num_executions** モニター・エレメントではなく **num_exec_with_metrics** モニター・エレメントを使用してください。**num_executions** モニター・エレメントでは、報告されるアクティビティー・メトリックにステートメントの実行が寄与したかどうかにかかわらず、ステートメントのすべての実行がカウントされるためです。

num_exec_with_metrics メトリックが収集された実行数 : モニター・エレメント

この SQL ステートメント・セクションが、メトリックが収集されて実行された回数。このエレメントを使用してパッケージ・キャッシュのステートメントにあるモニター・エレメントの実行ごとの値を計算することもできます。

表 677. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

num_extents_left プロセス残エクステント数 : モニター・エレメント

この表のリバランス・プロセス中に移動するエクステントで、まだ残っているエクステント数。

表 678. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_EXTENT_MOVEMENT_STATUS - エクステントの移動の進行状況メトリックの取得	常に収集される

num_extents_moved - 移動したエクステントの数 : モニター・エレメント

このエクステント移動操作中に、これまでに移動されたエクステントの数。

表 679. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_EXTENT_MOVEMENT_STATUS - エクステントの移動の進行状況メトリックの取得	常に収集される

num_gw_conn_switches - 接続切り替え回数

ある接続に対してエージェント・プールのエージェントがプライム状態にされ、別の DRDA データベースで使用するために再割り当てされた回数。

表 680. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

ほとんどのユーザーの場合、**num_poolagents** 構成パラメーターのデフォルト設定で最適なパフォーマンスが確保されます。この構成パラメーターのデフォルト設定では、自動的にエージェント・プールが管理され、エージェントが再割り当てされなくなります。

このモニター・エレメントの値を小さくするには、**num_poolagents** 構成パラメーターの値を調整してください。

num_indoubt_trans 未確定トランザクション数

データベース内に残っている未確定トランザクションの数。

表 681. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 未確定トランザクションは非コミット・トランザクションのログ・スペースを保留するため、ログがいっぱいになる可能性があります。ログがいっぱいになると、追加のトランザクションは完了できません。この問題を解決するには、手動による試行錯誤によって未確定トランザクションを解決する必要があります。このモニター・エレメントは、試行錯誤的に解決すべき未確定トランザクションの現在の残存数を提供します。

num_log_buffer_full - フル・ログ・バッファーの回数 : モニター・エレメント

エージェントが、ログ・レコードをログ・バッファーにコピーする際に、ログ・データをディスクに書き込むのを待つ回数。この値は、問題が生じるたびにエージェント数単位で大きくなります。例えば、バッファーがいっぱいの場合に 2 つのエージェントがログ・データをコピーしようとすると、この値は 2 単位ずつ大きくなります。

表 682. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE

表 682. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 683. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 684. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

このエレメントを使用して、 **logbufsz** データベース構成パラメーターの値を大きくする必要があるかどうかを判別します。

num_log_data_found_in_buffer ログ・データがバッファーにある回数

エージェントがバッファーからログ・データを読み取る回数。バッファーからログ・データを読み取る方が、ディスクから読み取るより速いので望ましいといえます。

表 685. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 686. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 このエレメントと *num_log_read_io* エレメントを組み合わせて使用すると、LOGBUFSZ データベース構成パラメーターの値を大きくする必要があるかどうかを判別します。

num_log_part_page_io 部分ログ・ページ書き込み数

ロガーが部分的なログ・データをディスクに書き込むのに発行した入出力要求の数。

表 687. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 688. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 このエレメントと *log_writes*、 *log_write_time*、および *num_log_write_io* エレメントを組み合わせて使用すると、現行ディスクがロギングに適しているかどうかを判別できます。

num_log_read_io ログ読み取り数

ロガーがログ・データをディスクから読み取るのに発行した入出力要求の数。

表 689. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 690. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 このエレメントと *log_reads* および *log_read_time* エレメントを組み合わせて使用すると、現行ディスクがロギングに適しているかどうかを判別できます。

num_log_write_io ログ書き込み数

ロガーがログ・データをディスクに書き込むのに発行した入出力要求の数。

表 691. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 692. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 このエレメントと *log_writes* および *log_write_time* エレメントを組み合わせて使用すると、現行ディスクがロギングに適しているかどうかを判別できます。

num_nodes_in_db2_instance パーティション内のノード数

スナップショットが取られたインスタンス上のノード数。

表 693. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

表 694. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
イベント・ログ・ヘッダー	event_log_header	-

使用法 このエレメントを使用して、インスタンスにおけるノード数を判別します。非パーティション・システムのデータベースの場合、この値は 1 になります。

num_remaps 再マップ数 : モニター・エレメント

このアクティビティーが再マップされた回数のカウント。 num_remaps がゼロより大きい場合、このアクティビティー・レコードの service_class_id は、アクティビティーが最後に再マップされたサービス・クラスの ID です。

表 695. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
WLM_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

表 696. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

使用法

この情報を使用して、予期された回数アクティビティーが再マップされたかどうかを検証します。

num_threshold_violations しきい値違反の回数 : モニター・エレメント

このデータベースが最後にアクティブにされてからそこで発生したしきい値違反の回数。

表 697. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 698. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法

このエレメントを使用すると、この特定のアプリケーションにおいてしきい値が有効であるかどうか、またはしきい値違反が多すぎないかを判別する助けになります。

num_transmissions 伝送回数

DB2 Connect ゲートウェイとこの DCS ステートメントを処理するのに使用されたホストの間の伝送回数。 (1 回のデータ伝送は、送信 1 回または受信 1 回を示します。)

注:

これは、DB2 UDB バージョン 8.1.2 以降には無関係なレガシー・モニター・エレメントです。 DB2 UDB バージョン 8.1.2 以降をご使用の場合は、**num_transmissions_group** モニター・エレメントを参照してください。

エレメント ID

num_transmissions

エレメント・タイプ

カウンター

表 699. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS ステートメント	dcs_stmt	ステートメント

使用法 このエレメントを使用すると、特定のステートメントの実行時間が長くかかった理由が明確になります。例えば、照会のときに大きな結果セットを戻すと、完了するまでにたくさんのデータ伝送回数が必要になります。

num_transmissions_group 伝送グループの回数

この DCS ステートメントを処理するのに使われた、 DB2 Connect ゲートウェイとホストの間の伝送範囲。 (1 回のデータ伝送は、送信 1 回または受信 1 回を示します。)

表 700. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS ステートメント	dcs_stmt	ステートメント

使用法 このエレメントを使用すると、特定のステートメントの実行時間が長くかかった理由が明確になります。例えば、照会のときに大きな結果セットを戻すと、完了するまでにたくさんのデータ伝送回数が必要になります。

伝送範囲を表す定数は以下のように記述され、sqlmon.h で定義されています。

API 定数	説明
SQLM_DCS_TRANS_GROUP_2	2 回の伝送
SQLM_DCS_TRANS_GROUP_3TO7	3 回から 7 回までの伝送
SQLM_DCS_TRANS_GROUP_8TO15	8 回から 15 回までの伝送
SQLM_DCS_TRANS_GROUP_16TO64	16 回から 64 回までの伝送
SQLM_DCS_TRANS_GROUP_GT64	64 回を超える伝送

number_in_bin ビン内の数 : モニター・エレメント

このエレメントは、ヒストグラム・ビンの中に入るアクティビティーまたは要求のカウント数を保持します。

表 701. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_histogrambin	-

使用法

このエレメントを使用すると、ヒストグラムのビンの高さを示すことができます。

olap_func_overflows OLAP 関数のオーバーフロー : モニター・エレメント

OLAP 関数データが、使用可能なソート・ヒープ・スペースを超えた回数。

表 702. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 703. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法

データベース・レベルでは、このエレメントと total_olap_funcs を組み合わせて使用すると、ディスクにオーバーフローした OLAP 関数のパーセンテージを計算できます。このパーセンテージが高く、OLAP 関数を使用するアプリケーションのパフォーマンスを改善する必要がある場合は、ソート・ヒープ・サイズを大きくすることを検討してください。

アプリケーション・レベルでは、このエレメントを使用すると、個々のアプリケーションにおける OLAP 関数のパフォーマンスを評価できます。

open_cursors オープン・カーソル数

アプリケーションで現在開いているカーソルの数。

表 704. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl	ステートメント

使用法 このエレメントを使用して、割り振られているメモリーの量を評価します。ターゲット・データベース上の DB2 クライアント、 DB2 Connect、またはデータベース・エージェントに割り振られるメモリー量は、現在開いているカーソルの数と関連しています。この情報は、キャパシティー・プランニングに利用できます。例えば、ブロッキングを行っている各オープン・カーソルのバッファー・サイズは RQRIOBLK です。 *deferred_prepare* を使用可能にすると、 2 つのバッファーが割り振られます。

このエレメントには、早期クローズによって閉じられたカーソルは組み込まれていません。ホスト・データベースが最後のレコードをクライアントに戻すと、早期クローズが生じます。ホストとゲートウェイではカーソルが閉じられますが、クライアント側では依然として開いています。早期クローズ・カーソルは、DB2 コール・レベル・インターフェースを使用して設定できます。

open_loc_curs 開かれているローカル・カーソル

このアプリケーション用に現在開かれているローカル・カーソルの数。 *open_loc_curs_blk* がカウントするカーソルを含みます。

エレメント ID

open_loc_curs

エレメント・タイプ

ゲージ

表 705. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

使用法 このエレメントと *open_loc_curs_blk* を組み合わせて使用すると、ローカル・ブロック・カーソルのパーセンテージを計算できます。パーセンテージが低い場合は、アプリケーション内の行ブロッキングを改善するとパフォーマンスが向上します。

リモート・アプリケーションが使用するカーソルについては、『*open_rem_curs*』を参照してください。

open_loc_curs_blk 開かれているローカル・ブロック・カーソル

このアプリケーション用に現在開かれているローカル・ブロック・カーソルの数。

エレメント ID

open_loc_curs_blk

エレメント・タイプ

ゲージ

表 706. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

使用法 このエレメントと *open_loc_curs* を組み合わせて使用すると、ローカル・ブロック・カーソルのパーセンテージを計算できます。パーセンテージが低い場合は、アプリケーション内の行ブロッキングを改善するとパフォーマンスが向上します。

- 未確定カーソル処理のレコード・ブロッキングについて、プリコンパイル・オプションをチェックする。
- カーソルを再定義してブロッキングを許可する (例えば、可能な場合は、カーソルに FOR FETCH ONLY を指定する)。

rej_curs_blk および acc_curs_blk が提供する情報を使用すると、構成パラメーターを調整して、アプリケーション内の行ブロッキングを改善できます。

リモート・アプリケーションが使用するブロック・カーソルについては、『*open_rem_curs_blk*』を参照してください。

open_rem_curs 開かれているリモート・カーソル

このアプリケーション用に現在開かれているリモート・カーソルの数。*open_rem_curs_blk* がカウントするカーソルを含みます。

エレメント ID

open_rem_curs

エレメント・タイプ

ゲージ

表 707. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

使用法 このエレメントと *open_rem_curs_blk* を組み合わせて使用すると、リモート・ブロック・カーソルのパーセンテージを計算できます。パーセンテージが低い場合は、アプリケーション内の行ブロッキングを改善するとパフォーマンスが向上します。詳しくは、『*open_rem_curs_blk*』を参照してください。

ローカル・データベースに接続されているアプリケーションが使用するオープン・カーソルの数については、『*open_loc_curs*』を参照してください。

open_rem_curs_blk 開かれているリモート・ブロック・カーソル

このアプリケーション用に現在開かれているリモート・ブロック・カーソルの数。

エレメント ID

open_rem_curs_blk

エレメント・タイプ

ゲージ

表 708. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

使用法 このエレメントと *open_rem_curs* を組み合わせて使用すると、リモート・ブロック・カーソルのパーセンテージを計算できます。パーセンテージが低い場合は、アプリケーション内の行ブロッキングを改善するとパフォーマンスが向上します。

- 未確定カーソル処理のレコード・ブロッキングについて、プリコンパイル・オプションをチェックする。
- カーソルを再定義してブロッキングを許可する (例えば、可能な場合は、カーソルに FOR FETCH ONLY を指定する)。

rej_curs_blk および *acc_curs_blk* が提供する情報を使用すると、構成パラメーターを調整して、アプリケーション内の行ブロッキングを改善できます。

ローカル・データベースに接続されているアプリケーションが使用するオープン・ブロック・カーソルの数については、『*open_loc_curs_blk*』を参照してください。

outbound_appl_id アウトバウンド・アプリケーション ID

アプリケーションが DRDA ホスト・データベースに接続すると、この ID が生成されます。この ID は、DB2 Connect ゲートウェイをホストに接続するときに使用しますが、**appl_id** モニター・エレメントはクライアントを DB2 Connect ゲートウェイに接続するときに使用します。

注: NetBIOS はサポートされなくなりました。 SNA とその API、APPC、APPN、および CPI-C もサポートされなくなりました。これらのプロトコルを使用している場合は、TCP/IP などのサポートされているプロトコルを使用して、ノードとデータベースを再カタログする必要があります。これらのプロトコルへの参照は、無視されることになります。

エレメント ID

outbound_appl_id

エレメント・タイプ

情報

表 709. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl_info	基本

使用法

このエレメントと **appl_id** を組み合わせて使用すると、アプリケーション情報のクライアントの部分とサーバーの部分を関連付けることができます。

この ID は、ネットワーク内でユニークな ID です。

ゲートウェイ・コンセントレーターがオンである場合、または DCS アプリケーションが作業論理単位内にない場合に、このエレメントはブランクになります。

形式 `Network.LU Name.Application instance`

例 `CAIBMTOR.OSFDBM0.930131194520`

詳細 このアプリケーション ID は、 APPC の会話が割り振られるとネットワーク上に流れる実 SNA LUWID (作業論理単位 ID) の表示可能な形式です。APPC が生成するアプリケーション ID は、ネットワーク名、LU 名、および LUWID インスタンス番号が連結されて構成され、これによってクライアント/サーバー・アプリケーションの固有のラベルが作成されます。ネットワーク名および LU 名に使用できる文字数は、それぞれ最大 8 文字です。アプリケーションのインスタンスは、 12 個の 10 進数文字を使用した LUWID インスタンス番号に対応します。

outbound_bytes_received 受信されたアウトバウンド・バイト数

DB2 Connect ゲートウェイがホストから受信したバイト数。ただし、通信プロトコルのオーバーヘッド (例えば TCP/IP や SNA のヘッダー) は含まれません。データ伝送レベルの場合: このデータ伝送回数を使用したすべてのステートメントの処理で、DB2 Connect ゲートウェイがホストから受信したバイト数。

表 710. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	基本
DCS アプリケーション	dcs_appl	基本
DCS ステートメント	dcs_stmt	ステートメント
データ伝送	stmt_transmissions	ステートメント

ステートメント・レベルでのスナップショット・モニターの場合、このカウンターはリセットできません。その他のレベルではこのカウンターはリセットできます。

使用法

このエレメントを使用して、ホスト・データベースから DB2 Connect ゲートウェイへのスループットを測定します。

outbound_bytes_received_bottom 受信された最小アウトバウンド・バイト数

このデータベースに対するステートメントまたはチェーン、またはこの DCS アプリケーション内のステートメントでまたはチェーン、このデータ伝送回数を使用したすべてのステートメントまたはチェーンを処理している間に、 DB2 Connect ゲートウェイがホストから受信したステートメントまたはチェーン単位の最小バイト数。

表 711. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データ伝送	stmt_transmissions	ステートメント

使用法 このエレメントは、 DB2 Connect ゲートウェイからホスト・データベースへのスループットを示すもう 1 つのパラメーターである「受信されたアウトバウンド・バイト数」と組み合わせて使用します。

outbound_bytes_received_top 受信された最大アウトバウンド・バイト数

DB2 Connect ゲートウェイがホストから受信したステートメントまたはチェーン単位の最大バイト数。この DCS データベースに対するステートメントまたはチェーン、またはこの DCS アプリケーション内のステートメントまたはチェーンを処理している間に、このデータ伝送回数を使用したすべてのステートメントまたはチェーンが対象になります。

表 712. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データ伝送	stmt_transmissions	ステートメント

使用法 このエレメントは、 DB2 Connect ゲートウェイからホスト・データベースへのスループットを示すもう 1 つのパラメーターである「受信されたアウトバウンド・バイト数」と組み合わせて使用します。

outbound_bytes_sent 送信されたアウトバウンド・バイト数

DB2 Connect ゲートウェイがホストに送信したバイト数。通信プロトコルのオーバーヘッド (例えば TCP/IP や SNA のヘッダー) は含まれません。データ伝送レベルの場合: このデータ伝送回数を使用したすべてのステートメントの処理で、DB2 Connect ゲートウェイがホストに送信したバイト数。

表 713. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	基本
DCS アプリケーション	dcs_appl	基本
DCS ステートメント	dcs_stmt	ステートメント
データ伝送	stmt_transmissions	ステートメント

ステートメント・レベルでのスナップショット・モニターの場合、このカウンターはリセットできません。その他のレベルではこのカウンターはリセットできます。

使用法 このエレメントを使用して、 DB2 Connect ゲートウェイからホスト・データベースへのスループットを測定します。

outbound_bytes_sent_bottom 送信された最小アウトバウンド・バイト数

DB2 Connect ゲートウェイがホストから受信したステートメントまたはチェーン単位の最小バイト数。この DCS データベースに対するステートメントまたはチェーン、またはこの DCS アプリケーション内のステートメントまたはチェーンを処理している間に、このデータ伝送回数を使用したすべてのステートメントまたはチェーンが対象になります。

表 714. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データ伝送	stmt_transmissions	ステートメント

使用法 このエレメントは、 DB2 Connect ゲートウェイからホスト・データベースへのスループットを示すもう 1 つのパラメーターである「送信されたアウトバウンド・バイト数」と組み合わせて使用します。

outbound_bytes_sent_top 送信された最大アウトバウンド・バイト数

このデータベースに対するステートメントまたはチェーン、またはこの DCS アプリケーション内のステートメントまたはチェーンで、このデータ伝送回数を使用したすべてのステートメントまたはチェーンを処理している間に、 DB2 Connect ゲートウェイがホストに送信したステートメントまたはチェーン単位の最大バイト数。

表 715. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データ伝送	stmt_transmissions	ステートメント

使用法 このエレメントは、 DB2 Connect ゲートウェイからホスト・データベースへのスループットを示すもう 1 つのパラメーターである「送信されたアウトバウンド・バイト数」と組み合わせて使用します。

outbound_comm_address アウトバウンド通信アドレス

これはターゲット・データベースの通信アドレスです。例えば、SNA ネット ID と LU パートナー名、または TCP/IP 用の IP アドレスとポート番号などです。

エレメント ID

outbound_comm_address

エレメント・タイプ

情報

表 716. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
DCS アプリケーション	dcs_appl_info	基本

使用法 このエレメントは、DCS アプリケーションに関する問題判別に使用します。

outbound_comm_protocol アウトバウンド通信プロトコル

DB2 Connect ゲートウェイとホストの間で使用される通信プロトコル。

表 717. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl_info	基本

使用法

このエレメントは、DCS アプリケーションに関する問題判別に使用します。有効な値は、次のとおりです。

- SQLM_PROT_TCPIP

outbound_sequence_no アウトバウンド・シーケンス番号

ゲートウェイ・コンセントレーターがオンである場合、または DCS アプリケーションが作業論理単位内にない場合に、このエレメントはブランクになります。

表 718. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl_info	基本

overflow_accesses - オーバーフロー・レコードへのアクセス : モニター・エレメント

この表のオーバーフローした行へのアクセス (読み取りおよび書き込み) 数。

表 719. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLE 表関数 - 表メトリックの取得	常に収集される

表 720. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 721. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表	event_table	-

使用法

オーバーフローした行は、データのフラグメント化が生じたことを示します。この数値が大きい場合は、このフラグメント化をクリーンアップする REORG ユーティリティーを使用して表を再編成することによって、表のパフォーマンスを改善できることがあります。

行が更新されて、以前に書き込まれていたデータ・ページに収まらなくなった場合に、行はオーバーフローします。 VARCHAR または ALTER TABLE ステートメントを更新すると、こうしたことが起こります。

overflow_creates - オーバーフローの作成 : モニター・エレメント

この表で作成された、オーバーフローした行の数。

表 722. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	常に収集される

使用法

package_name - パッケージ名 : モニター・エレメント

現在実行中の SQL ステートメントが含まれているパッケージの名前。

表 723. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	常に収集される

表 724. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント
DCS ステートメント	dcs_stmt	ステートメント

表 725. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック[1]	event_detailed_dlconn	-
ステートメント	event_stmt	-
アクティビティー	event_activitystmt	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

このエレメントを使用すると、実行中のアプリケーション・プログラムや SQL ステートメントを識別できます。

package_schema - パッケージ・スキーマ : モニター・エレメント

アクティビティーが SQL ステートメントの場合、これはパッケージのスキーマ名を表します。

表 726. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	常に収集される

表 727. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-

package_version_id - パッケージ・バージョン : モニター・エレメント

特定のパッケージ名および作成者について、複数のバージョンが存在する場合があります (DB2 バージョン 8 以降)。パッケージ・バージョンは、現在実行中の SQL ステートメントを含むパッケージのバージョン ID を示します。パッケージのバージョンは、組み込み SQL プログラムをプリコンパイル (PREP) するときに VERSION キーワードを使用して決めます。プリコンパイル時に指定がない場合は、パッケージ・バージョンが "" (空ストリング) となります。

表 728. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	常に収集される

表 729. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント

表 730. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
ステートメント	event_stmt	-
アクティビティー	event_activitystmt	-

使用法

このエレメントを使用して、パッケージおよび現在実行中の SQL ステートメントの識別に役立てることができます。

page_allocations - ページ割り振り : モニター・エレメント

索引に割り振られたページ数。

表 731. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

page_reorgs ページ再編成

表のページ再編成が実行された回数。

表 732. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 733. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表	event_table	-

使用法

以下の状態では、ページに十分なスペースがあってもフラグメント化されることがあります。

- 新しい行が挿入される場合
- 既存の行が更新され、その結果レコード・サイズが大きくなる場合

ページがフラグメント化されている場合は、ページの再編成が必要になることがあります。再編成すると、フラグメント化されたすべてのスペースを連続したエリアに移動して、そこに新しいレコードが書き込まれます。このようなページの再編成 (page reorg) の実行には、数千の指示が必要になることがあります。また、操作のログ・レコードも生成されます。

ページ再編成の回数が多すぎると、最適な挿入パフォーマンスを達成できないことがあります。 REORG TABLE ユーティリティーを使用すると、表を再編成してフラグメント化をなくすことができます。さらに、ALTER TABLE ステートメントで APPEND パラメーターを使用すると、表の末尾にすべての挿入を付加するように指定でき、ページの再編成を回避できます。

行の更新をすると行の長さが増えるような場合は、そのページに新しい行を挿入するだけの場所があっても、そのスペースのフラグメント化を解消する

ためにページ REORG が必要になる場合があります。ページに新しい大きな行を挿入するだけの場所がない場合は、オーバーフロー・レコードが作成されて、読み取りのときに *overflow_accesses* の原因となります。どちらの状態も、可変長列の代わりに固定長列を使用することで回避できます。

pages_from_block_ios - ブロック入出力によって読み取られたページ数の合計 : モニター・エレメント

ブロック入出力でバッファー・プールのページ・エリアに読み取られたページ数の合計。

表 734. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 735. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	bufferpool	バッファー・プール

使用法

ブロック・ベースのバッファー・プールが使用可能になっている場合、このエレメントは、ブロック入出力によって読み取られたページ数の合計を報告します。そうでない場合、このエレメントは 0 を戻します。

ブロック・ベースの入出力ごとに順次プリフェッチされたページ数の平均を計算するには、**pages_from_block_ios** モニター・エレメントの値を **block_ios** モニター・エレメントの値で除算します。この値が、CREATE BUFFERPOOL または ALTER BUFFERPOOL ステートメントでブロック・ベースのバッファー・プールについて定義した BLOCKSIZE オプションより大幅に小さい場合は、ブロック・ベースの入出力の利点が最大限に活用されていません。考えられる原因の 1 つは、順次プリフェッチされている表スペースのエクステント・サイズと、ブロック・ベースのバッファー・プールのブロック・サイズが一致していないことです。

pages_from_vectored_ios - ベクトル化入出力によって読み取られたページ数の合計 : モニター・エレメント

ベクトル化入出力でバッファー・プールのページ・エリアに読み取られたページ数の合計。

表 736. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 737. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	bufferpool	バッファー・プール

pages_merged - マージされたページ : モニター・エレメント

マージされた索引ページの数。

表 738. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

pages_read - 読み取られたページの数 : モニター・エレメント

REGULAR 表スペースおよび LARGE 表スペースの物理表スペース・コンテナーから読み取られたページ (データ、索引、および XML) の数。

表 739. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE

使用法

pages_written - 書き込まれたページの数 : モニター・エレメント

表スペース・コンテナーに物理的に書き込まれたページ (データ、索引、および XML) の数。

表 740. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE

使用法

parent_activity_id 親アクティビティー ID : モニター・エレメント

アクティビティーの親アクティビティーの作業単位内における、その親アクティビティーのユニーク ID。親アクティビティーがない場合、このモニター・エレメントの値は 0 です。

表 741. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

表 742. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

使用法

このエレメントを **parent_uow_id** エレメントおよび **appl_id** エレメントと組み合わせて使用すると、このアクティビティー・レコードで記述されているアクティビティーの親アクティビティーを一意的に識別できます。

parent_uow_id 親作業単位 ID : モニター・エレメント

アプリケーション・ハンドル内の固有の作業単位 ID。アクティビティーの親アクティビティーが発生する作業単位の ID。親アクティビティーがない場合、値は 0 です。

表 743. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

表 744. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

使用法

このエレメントを **parent_activity_id** エレメントおよび **appl_id** エレメントと組み合わせて使用すると、このアクティビティー・レコードで記述されているアクティビティーの親アクティビティーを一意的に識別できます。

partial_record 部分レコード : モニター・エレメント

イベント・モニター・レコードが部分レコードでしかないことを示します。

表 745. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
表	event_table	-
表スペース	event_tablespace	-
バッファー・プール	event_bufferpool	-
接続	event_conn	-
ステートメント	event_stmt	-
ステートメント	event_subsection	-
トランザクション	event_xact	-
アクティビティー	event_activity	-

使用法

ほとんどのイベント・モニターは、データベースが非活動状態になるまでそれぞれの結果を出力しません。 FLUSH EVENT MONITOR <monitorName> ステートメントを使用すると、モニター値をイベント・モニター出力ライターに強制的に渡すことができます。これにより、イベント・モニターを停止して再始動する必要なしに、イベント・モニター・レコードをライターに強制的に渡すことができます。このエレメントは、イベント・モニター・レコードがフラッシュ操作の結果であり、したがってそれが部分レコードであるかどうかを示します。

イベント・モニターのフラッシュが原因で値がリセットされることはありません。これは、イベント・モニターが起動するときに、完全イベント・モニター・レコードが生成されることを意味します。

event_activity 論理データ・グループで、**partial_record** モニター・エレメントの有効値は以下のとおりです。

0 アクティビティーの終わりに通常どおりアクティビティー・レコードが生成されました。

1 WLM_CAPTURE_ACTIVITY_IN_PROGRESS ストアード・プロシージャーの呼び出しの結果としてアクティビティー・レコードが生成されました。

2 使用可能なストレージが足りずにレコードを作成できなかったため、このアクティビティーに関する情報がありません。
event_activity、event_activitystmt、または event_activityvals レコードの情報が失われている可能性があります。

participant_no デッドロック内の参加者

このデッドロック内のこの参加者を一意的に識別するシーケンス番号。

エレメント ID

participant_no

エレメント・タイプ
情報

表 746. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	event_dlconn	-
詳細付きデッドロック	event_detailed_dlconn	-

使用法 モニター・アプリケーションでこのエレメントを使用すると、デッドロック接続イベント・レコードとデッドロック・イベント・レコードを関連付けることができます。

participant_no_holding_lk アプリケーションが必要とするオブジェクトのロックを保留する参加者

このアプリケーションがロックの取得を待機しているオブジェクトのロックを保留しているアプリケーションの参加者番号。

エレメント ID
participant_no_holding_lk

エレメント・タイプ
情報

表 747. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	event_dlconn	-
詳細付きデッドロック	event_detailed_dlconn	-

使用法 このエレメントは、リソースの競合状態にあるアプリケーションの判別に利用できます。

partition_number パーティション番号

このエレメントは、パーティション・データベース環境で、表書き込みイベント・モニターによってターゲット SQL 表でのみ使用されます。この値はイベント・モニターのデータが挿入されたパーティションの番号を示します。

表 748. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
-	-	-

passthru_time パススルー時間

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時か、またはデータベース・モニター・カウンターの最後のリセット時のどちらか最近の方の時点以降に、このフェデレーテッド・サーバー・インスタンス上で実行されているすべてのアプリケーションまたは単一アプリケーションからの PASSTHRU ステートメントに対して、このデータ・ソースが応答に要した合計時間が含まれていま

す (ミリ秒単位)。応答時間は、フェデレーテッド・サーバーが PASSTHRU ステートメントをデータ・ソースにサブミットしてからデータ・ソースが PASSTHRU ステートメントを処理したことを応答するまでの時間です。

表 749. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	タイム・スタンプ
アプリケーション	appl_remote	タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

このエレメントを使用すると、このデータ・ソースでステートメントをパススルー・モードで処理するのに要した実際の時間を判別できます。

passthrus パススルー数

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時か、またはデータベース・モニター・カウンターの最後のリセット時のどちらか最近の方の時点以降に、いずれかのアプリケーションに代わってフェデレーテッド・サーバーがこのデータ・ソースに直接渡した SQL ステートメントの合計数が含まれています。

表 750. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、フェデレーテッド・サーバーがネイティブに処理できる SQL ステートメントのパーセンテージ、およびパススルー・モードが必要なパーセンテージを判別できます。この値が大きい場合は、原因を突き止めて、ネイティブ・サポートをさらに効率的に使用する方法を検討してください。

piped_sorts_accepted 受け入れられたパイプ・ソート

受け入れられたパイプ・ソートの数。

表 751. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 システム上のアクティブ・ソートには、それぞれメモリーが割り振られるため、使用可能なシステム・メモリーからソートのためのメモリー量を多く取り過ぎることがあります。

パイプ・ソートを要求した数に対して、受け入れられたパイプ・ソートの数が少ない場合は、次の構成パラメーターのいずれかまたは両方を調整するとソート・パフォーマンスを改善できます。

- sortheap
- sheapthres

パイプ・ソートがリジェクトされる場合は、ソート・ヒープを少なくするか、またはソート・ヒープしきい値を大きくすることを考慮してください。ただし、これらのオプションが与えるそれぞれの影響を知っておく必要があります。ソート・ヒープしきい値を高くすると、ソート処理に割り振られるメモリーの量が多くなる可能性があります。その場合、メモリーがディスクにページングされることがあります。ソート・ヒープを少なくすると、マージ・フェーズが増えてソート処理が遅くなることがあります。

piped_sorts_requested 要求されたパイプ・ソート数

要求されたパイプ・ソートの数。

表 752. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 システム上のアクティブ・ソートには、それぞれメモリーが割り振られるため、使用可能なシステム・メモリーからソートのためのメモリー量を多く取り過ぎることがあります。

ソート・リスト・ヒープ (*sortheap*) およびソート・ヒープしきい値 (*sheapthres*) の構成パラメーターは、ソート操作に使用するメモリー量をコントロールするのに利用できます。また、これらのパラメーターを使用すると、ソートをパイプ化するかどうかを決定することもできます。

ソートをパイプ化するとディスク入出力が少なくなり、パイプ・ソートの数を増やせるので、ソート操作のパフォーマンスを改善し、さらにはシステム全体のパフォーマンスもよくなります。ただし、ソート・ヒープをソートに割り振る時にソート・ヒープしきい値を超えてしまうと、パイプ・ソートは受け入れてもらえません。パイプ・ソートがリジェクトされる場合については、『*piped_sorts_accepted*』を参照してください。

SQL EXPLAIN 出力には、オプティマイザーがパイプ・ソートを要求するかどうかが示されます。

pkg_cache_inserts パッケージ・キャッシュ挿入

要求したセクションが使用できなかったためにパッケージ・キャッシュへのロードが必要になった回数の合計。このカウントには、システムが暗黙に準備した数が含まれます。

エレメント ID

pkg_cache_inserts

エレメント・タイプ

カウンター

表 753. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 754. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 「パッケージ・キャッシュ検索」と組み合わせると、次の公式を使用してパッケージ・キャッシュ・ヒット率を計算できます。

```
1 - (パッケージ・キャッシュ挿入 / パッケージ・キャッシュ検索)
```

このエレメントの使用法については、『pkg_cache_lookups』を参照してください。

pkg_cache_lookups パッケージ・キャッシュ検索

アプリケーションがパッケージ・キャッシュ内のセクションやパッケージを検索した回数。データベース・レベルでは、データベースの開始以降、またはモニター・データのリセット以降の参照の合計数を示します。このカウンターには、セクションをキャッシュにすでにロードしてある場合と、セクションをキャッシュにロードする必要がある場合が含まれます。エージェントがさまざまなアプリケーションと関連付けられているようなコンセントレーター環境では、新しいエージェントに必要なセクションやパッケージがローカル・ストレージ内にない場合に、パッケージ・キャッシュの検索がさらに必要になります。

表 755. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 756. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法

パッケージ・キャッシュ・ミス率の計算には次の数式を使用します。

```
1 - (パッケージ・キャッシュ挿入 / パッケージ・キャッシュ検索)
```

パッケージ・キャッシュ・ミス率は、パッケージ・キャッシュが効果的に利用されているかどうかを示します。このミス率が低い場合 (0.2 未満)、キャッシュは効果的に動作しています。このミス率が高い場合は、パッケージ・キャッシュを大きくする必要があることを示します。

パッケージ・キャッシュのサイズを変えて試すことにより、 *pckcachesz* 構成パラメーターに最適な値を見つける必要があります。例えば、キャッシュのサイズを小さくしても *pkg_cache_inserts* エレメントが増えない場合は、パッケージ・キャッシュのサイズをさらに小さくできます。パッケージ・キャッシュのサイズを小さくすれば、その分のシステム・リソースを他の作業のために使えるようになります。または、pkg_cache_inserts の数を少なくして、パッケージ・キャッシュのサイズを大きくすると、システム全体のパフォーマンスを向上させることができます。この実験は、フル・ワークロードの条件で行うのが最善です。

このエレメントと *ddl_sql_stmts* を組み合わせて使用すると、 DDL ステートメントを実行したときにパッケージ・キャッシュのパフォーマンスに影響を与えるかどうかを判別できます。 DDL ステートメントを実行すると、動的 SQL ステートメントの一部のセクションが無効になる場合があります。無効なセクションは、次に使用されるときにシステムが暗黙的に準備します。 DDL ステートメントを実行すると、多数のセクションが無効になり、こうしたセクションを準備するときに余分に必要になるオーバーヘッドのためにパフォーマンスが大きく低下することがあります。この場合のパッケージ・キャッシュ・ヒット率は、無効なセクションの暗黙的な再コンパイルを反映します。キャッシュに挿入される新しいセクションは反映されないので、パッケージ・キャッシュのサイズを大きくしても総合的なパフォーマンスは改善できません。フル環境を対象に作業する前に、アプリケーション自体のキャッシュを調整すれば、混乱を避けることができます。

実行すべきアクションを考える前に、パッケージ・キャッシュ・ヒット率の値に DDL ステートメントがどのような役割を果たしているのかを明確にする必要があります。 DDL ステートメントがあまり発生しない場合は、キャッシュのサイズを大きくするとキャッシュのパフォーマンスを改善できる場合があります。 DDL ステートメントが頻繁に使用される場合は、 DDL ステートメントを制限する (時間を限定するなど) と改善できる場合があります。

static_sql_stmts および *dynamic_sql_stmts* のカウントは、キャッシュに入れるセクションの数量とタイプに関する情報を提供するときに利用できます。

「パッケージ・キャッシュ・サイズ (*pckcachesz*)」構成パラメーターについて詳しくは、「*管理ガイド*」を参照してください。

注: この情報をデータベース・レベルで使用すると、すべてのアプリケーションについて個別の平均パッケージ・キャッシュ・ヒット率を計算できます。特定のアプリケーションのパッケージ・キャッシュ・ヒット率が知りたいときには、この情報をアプリケーション・レベルで調べてください。実行頻度の少ないアプリケーションのキャッシュ要件を満たすためにパッケージ・キャッシュのサイズを大きくしてもあまり意味がありません。

pkg_cache_num_overflows パッケージ・キャッシュ・オーバーフロー

割り振られたメモリーの境界からパッケージ・キャッシュがオーバーフローした回数。

表 757. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 758. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法

このエレメントと **pkg_cache_size_top** モニター・エレメントを組み合わせて使用すると、オーバーフローを回避するのにパッケージ・キャッシュのサイズを大きくする必要があるかどうかを判別できます。

pkg_cache_size_top パッケージ・キャッシュの最高水準点

パッケージ・キャッシュが到達した最大サイズ。

注: pkg_cache_size_top モニター・エレメントは、DB2 バージョン 9.5 以降では非推奨になっています。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 759. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 760. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法

パッケージ・キャッシュがオーバーフローした場合、このエレメントは、オーバーフロー時にパッケージ・キャッシュが到達した最大サイズになります。

このような状態が発生したかどうかを確認するには、 **pkg_cache_num_overflows** モニター・エレメントをチェックしてください。

ワークロードに必要なパッケージ・キャッシュの最小サイズは次のように決定できます。

```
パッケージ・キャッシュの最大サイズ / 4096
```

この結果を切り上げた整数が、オーバーフローを避けるためにパッケージ・キャッシュが必要とする 4K ページの最小数になります。

pool_async_data_read_reqs - バッファー・プール非同期読み取り要求 : モニター・エレメント

プリフェッチャーによるオペレーティング・システムに対する非同期読み取り要求の数。これらの要求は一般に、複数ページから成るラージ・ブロック入出力です。

表 761. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 762. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 763. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
表スペース	event_tablespace	-

使用法

読み取り要求ごとのデータ・ページの平均数を計算するには、次の数式を使用します。

```
pool_async_data_reads / pool_async_data_read_reqs
```

この平均値は、プリフェッチャーで使用される読み取り入出力平均サイズを判別するのに役立ちます。また、測定対象ワークロードのラージ・ブロック入出力要件を理解するうえでもこのデータは役立ちます。

プリフェッチャー読み取り入出力の最大サイズは、関係する表スペースの CREATE TABLESPACE ステートメントの EXTENTSIZE オプションで指定された値ですが、次のようないくつかの状況下ではそれよりも小さくなることがあります。

- エクステントのいくつかのページがすでにバッファー・プールに入っている場合
- オペレーティング・システムの能力を超えている場合
- EXTENTSIZE オプションの値が非常に大きく、ラージ入出力を実行すると全体のパフォーマンスに悪影響が出る場合

pool_async_data_reads バッファー・プール非同期データ読み取り : モニター・エレメント

すべてのタイプの表スペースについて、非同期エンジン・ディスパッチ可能単位 (EDU) によって表スペース・コンテナー (物理) から読み取られた、データ・ページの数を示します。

表 764. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 765. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 766. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
表スペース	event_tablespace	-

使用法

このエレメントと **pool_data_p_reads** を組み合わせて使用すると、同期的に実行された物理読み取り数を計算できます (つまり、データベース・マネージャーのエージェントが実行したデータ・ページ物理読み取り数)。次の数式を使用します。

```
 1-((pool_data_p_reads+pool_index_p_reads)-(pool_async_data_reads+pool_async_index_reads))
/(pool_data_l_reads+pool_index_l_reads)
```

非同期読み取り数と同期読み取り数を比較すると、プリフェッチャーの動作状態がわかります。このエレメントは、**num_ioservers** 構成パラメーターを調整するときに役に立ちます。

非同期読み取りは、データベース・マネージャーのプリフェッチャーが実行します。

pool_async_data_writes - バッファー・プール非同期データ書き込み : モニター・エレメント

非同期ページ・クリーナーまたはプリフェッチャーによりバッファー・プール・データ・ページがディスクに物理的に書き込まれた回数。プリフェッチャーは、プリフェッチするページの場所を作るために、ダーティー・ページをディスクに書き込む場合があります。

表 767. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 768. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 769. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
表スペース	event_tablespace	-

使用法 このエレメントを **buffer_pool_data_writes** モニター・エレメントと一緒に使用すると、同期的に実行された物理書き込み要求の数 (つまり、データベース・マネージャーのエージェントが実行したデータ・ページの物理書き込み数) を計算できます。次の数式を使用します。

```
pool_data_writes - pool_async_data_writes
```

非同期読み取り数と同期読み取り数を比較すると、バッファー・プール・ページ・クリーナーの動作状態がわかります。この比率は、 **num_iocleaners** 構成パラメーターを調整するときに役に立ちます。

pool_async_index_read_reqs - バッファー・プール非同期索引読み取り要求 : モニター・エレメント

索引ページの非同期読み取り要求の数。

表 770. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 771. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 772. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
表スペース	event_tablespace	-

使用法 非同期要求当たりの索引ページ読み取りの数を計算するには、次の数式を使用します。

```
pool_async_index_reads / pool_async_index_read_reqs
```

この平均値は、プリフェッチャーとのそれぞれの対話で索引ページに関して行われる非同期入出力量を判別するのに役立ちます。

pool_async_index_reads - バッファー・プール非同期索引読み取り : モニター・エレメント

すべてのタイプの表スペースについて、非同期エンジン・ディスパッチ可能単位 (EDU) によって表スペース・コンテナー (物理) から読み取られた、索引ページの数を示します。

表 773. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 774. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 775. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
表スペース	event_tablespace	-

使用法

このエレメントを **pool_index_p_reads** モニター・エレメントと一緒に使用すると、同期的に実行された物理読み取りの数 (つまり、データベース・マネージャーのエージェントが実行した索引ページ物理読み取り数) を計算できます。次の数式を使用します。

```
1 - ((pool_data_p_reads + pool_index_p_reads) - (pool_async_data_reads + pool_async_index_reads))
/ (pool_data_l_reads + pool_index_l_reads)
```

非同期読み取り数と同期読み取り数を比較すると、プリフェッチャーの動作状態がわかります。このエレメントは、**num_ioservers** 構成パラメーターを調整するときに役に立ちます。

非同期読み取りは、データベース・マネージャーのプリフェッチャーが実行します。

pool_async_index_writes - バッファー・プール非同期索引書き込み : モニター・エレメント

非同期ページ・クリーナーまたはプリフェッチャーによりバッファー・プール索引ページがディスクに物理的に書き込まれた回数。プリフェッチャーは、プリフェッチするページの場所を作るために、ダーティー・ページをディスクに書き込む場合があります。

表 776. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 777. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 778. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
表スペース	event_tablespace	-

使用法

このエレメントを **pool_index_writes** モニター・エレメントと一緒に使用すると、同期的に実行された物理索引書き込み要求の数 (つまり、データベース・マネージャーのエージェントが実行した索引ページ物理書き込み数) を計算できます。次の数式を使用します。

```
pool_index_writes - pool_async_index_writes
```

非同期読み取り数と同期読み取り数を比較すると、バッファー・プール・ページ・クリーナーの動作状態がわかります。この比率は、 **num_iocleaners** 構成パラメーターを調整するときに役に立ちます。

pool_async_read_time バッファー・プール非同期読み取り時間

すべてのタイプの表スペースについて、非同期エンジン・ディスパッチ可能単位 (EDU) によって表スペース・コンテナー (物理) からデータ・ページおよび索引ページを読み取るために費やされた合計時間を示します。この値はミリ秒単位で示されます。

エレメント ID

pool_async_read_time

エレメント・タイプ

カウンター

表 779. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	バッファー・プール	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 780. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
表スペース	event_tablespace	-

使用法 このエレメントを使用すると同期読み取りの経過時間を計算できます。次の数式を使用します。

```
pool_read_time - pool_async_read_time
```

このエレメントを使用すると、平均非同期読み取り時間も計算できます。次の数式を使用します。

```
pool_async_read_time / pool_async_data_reads
```

これらの計算は、実行中の入出力操作を把握するときに使用します。

pool_async_write_time バッファー・プール非同期書き込み時間

データベース・マネージャーのページ・クリーナーがデータ・ページまたは索引ページをバッファー・プールからディスクに書き込むのに要した合計経過時間。

エレメント ID

pool_async_write_time

エレメント・タイプ

カウンター

表 781. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	バッファー・プール	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 782. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
表スペース	event_tablespace	-

使用法 同期によるページ書き込みでの経過時間を計算するには、次の数式を使用します。

```
pool_write_time - pool_async_write_time
```

このエレメントを使用すると、平均非同期読み取り時間も計算できます。次の数式を使用します。

```
pool_async_write_time
 / (pool_async_data_writes
   + pool_async_index_writes)
```

これらの計算は、実行中の入出力操作を把握するときに使用します。

pool_async_xda_read_reqs - バッファー・プール非同期 XDA 読み取り要求 : モニター・エレメント

XML ストレージ・オブジェクト (XDA) データの非同期読み取り要求の数。

表 783. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 784. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 785. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
表スペース	event_tablespace	-

使用法 非同期要求当たりの XML ストレージ・オブジェクト・データ・ページ読み取りの平均数を計算するには、次の数式を使用します。

```
pool_async_xda_reads / pool_async_xda_read_reqs
```

この平均値は、各プリフェッチャーとのやりとりでの非同期入出力量を判別するのに利用します。

pool_async_xda_reads - バッファー・プール非同期 XDA データ読み取り : モニター・エレメント

すべてのタイプの表スペースについて、非同期エンジン・ディスパッチ可能単位 (EDU) によって表スペース・コンテナー (物理) から読み取られた、XML ストレージ・オブジェクト (XDA) データ・ページの数を示します。

表 786. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 787. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 788. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
表スペース	event_tablespace	-

使用法

pool_async_xda_reads および **pool_xda_p_reads** モニター・エレメントを使用して、XML ストレージ・オブジェクト・データ・ページ上で同期的に実行された物理読み取りの数 (つまり、データベース・マネージャーのエージェントが XML データに対して実行したデータ・ページの物理読み取り数) を計算します。次の数式を使用します。

```
pool_xda_p_reads - pool_async_xda_reads
```

非同期読み取り数と同期読み取り数を比較すると、プリフェッチャーの動作状態がわかります。このエレメントは、**num_ioservers** 構成パラメーターを調整するときに役に立ちます。

非同期読み取りは、データベース・マネージャーのプリフェッチャーが実行します。

pool_async_xda_writes - バッファー・プール非同期 XDA データ書き込み : モニター・エレメント

非同期ページ・クリーナーまたはプリフェッチャーにより、XML ストレージ・オブジェクト (XDA) のバッファー・プール・データ・ページがディスクに物理的に書き込まれた回数。プリフェッチャーは、プリフェッチするページの場所を作るために、ダーティー・ページをディスクに書き込む場合があります。

表 789. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 790. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 791. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
表スペース	event_tablespace	-

使用法 このエレメントを **pool_xda_writes** モニター・エレメントと一緒に使用すると、XML ストレージ・オブジェクト・データ・ページ上で同期的に実行された物理書き込み要求の数 (つまり、データベース・マネージャーのエージェントが XML データに対して実行したデータ・ページの物理書き込み数) を計算できます。次の数式を使用します。

```
pool_xda_writes - pool_async_xda_writes
```

非同期読み取り数と同期読み取り数を比較すると、バッファー・プール・ページ・クリーナーの動作状態がわかります。この比率は、 **num_iocleaners** 構成パラメーターを調整するときに役に立ちます。

pool_config_size メモリー・プールの構成済みサイズ

DB2 データベース・システム内のメモリー・プールの内部構成済みのサイズです。

表 792. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	memory_pool	基本

表 792. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	memory_pool	基本
アプリケーション	memory_pool	基本

表 793. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_dbmemuse	-
接続	event_connmemuse	-

使用法 システム・メモリーの使用量を追跡するときに、この値と *pool_cur_size*、*pool_id*、および *pool_watermark* を組み合わせて使用します。

メモリー・プールがほぼ満杯状態になっているかどうかを確認するには、*pool_config_size* と *pool_cur_size* を比較します。例えば、ユーティリティー・ヒープが小さすぎるとします。この問題は、一定間隔でスナップショットを取り、スナップショット出力のユーティリティー・ヒープ・セクションを調べることによって診断できます。必要な場合は、メモリー不足の障害が起きないように、 *pool_cur_size* を *pool_config_size* より大きくすることもできます。大きくなる頻度が非常に少ない場合は、おそらく追加のアクションは必要ありません。しかし、常に *pool_cur_size* の値が *pool_config_size* の値に近いか大きい場合は、ユーティリティー・ヒープのサイズを大きくすることを考慮してください。

pool_cur_size メモリー・プールの現行サイズ

メモリー・プールの現行サイズ。

表 794. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	memory_pool	基本
データベース	memory_pool	基本
アプリケーション	memory_pool	基本

表 795. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_dbmemuse	-
接続	event_connmemuse	-

使用法 システム・メモリーの使用量を追跡するときに、この値と *pool_config_size*、 *pool_id*、および *pool_watermark* を組み合わせて使用します。

メモリー・プールがほぼ満杯状態になっているかどうかを確認するには、*pool_config_size* と *pool_cur_size* を比較します。例えば、ユーティリティー・ヒープが小さすぎるとします。この問題は、一定間隔でスナップショットを取り、スナップショット出力のユーティリティー・ヒープ・セクション

を調べることによって診断できます。 *pool_cur_size* の値が *pool_config_size* の値に近い場合は、ユーティリティー・ヒープのサイズを大きくすることを考慮してください。

pool_data_l_reads - バッファー・プール・データの論理読み取り : モニター・エレメント

REGULAR 表スペースおよび LARGE 表スペースのバッファー・プール (論理) から要求されたデータ・ページの数。

表 796. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 797. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール

表 797. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 798. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
表スペース	event_tablespace	-
接続	event_conn	-
ステートメント	event_stmt	-
アクティビティー	event_activity	バッファー・プール、ステートメント

使用法

このカウントには、次のデータへのアクセスが含まれます。

- データベース・マネージャーがページの処理を必要としたときにバッファー・プールにすでにあるデータ。
- データベース・マネージャーがページを処理する前にバッファー・プールに読み取られたデータ。

pool_data_l_reads および **pool_data_p_reads** モニター・エレメントを使用して、次の数式でバッファー・プールの全体的なデータ・ページ・ヒット率を計算します。

```
1 - ((pool_data_p_reads - pool_async_data_reads) / pool_data_l_reads)
```

バッファー・プール・サイズを大きくすると、一般的にヒット率は高くなりますが、ある点を超えると逆に低くなります。理想的には、データベース全体を保管できるような大きなバッファー・プールを割り振ることができれば、システムが稼働中のヒット率は 100% になります。しかし、現実的にはそうしたことは起こりません。実際には、使用するデータのサイズとそのデータへのアクセス方法によってヒ

ット率の重要度は異なります。非常に大きなデータベースでアクセスが均等な場合は、ヒット率が低くなります。表が非常に大きな場合は、対応する方法はほとんどありません。

頻繁にアクセスされる小さめの表と索引のヒット率を上げるには、そうした表と索引を個別のバッファー・プールに割り当ててください。

pool_data_p_reads - バッファー・プール・データの物理読み取り : モニター・エレメント

REGULAR 表スペースおよび LARGE 表スペースの表スペース・コンテナー (物理) から読み取られた、データ・ページの数を示します。

表 799. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 800. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 801. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
表スペース	event_tablespace	-
接続	event_conn	-
ステートメント	event_stmt	-
アクティビティー	event_activity	バッファー・プール、ステートメント

使用法

このエレメントを **pool_data_l_reads** および **pool_async_data_reads** モニター・エレメントと一緒に使用して、同期的に実行された物理読み取りの数 (つまり、データベース・マネージャーのエージェントが実行したデータ・ページの物理読み取り数) を計算します。次の数式を使用します。

```
1 - ((pool_data_p_reads + pool_index_p_reads) - (pool_async_data_reads + pool_async_index_reads))
 / (pool_data_l_reads + pool_index_l_reads)
```

非同期読み取り数と同期読み取り数を比較すると、プリフェッチャーの動作状態がわかります。この情報は、**num_ioservers** 構成パラメーターを調整するときに役立つ場合があります。

pool_data_writes - バッファー・プールへのデータの書き込み : モニター・エレメント

バッファー・プールのデータ・ページが物理的にディスクに書き込まれた回数。

表 802. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 803. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 804. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
表スペース	event_tablespace	-
接続	event_conn	-

使用法

pool_data_p_reads モニター・エレメントの値のパーセンテージが高いためにバッファー・プールのデータ・ページがディスクに書き込まれる場合は、データベースで利用可能なバッファー・プール・ページ数を増やすとパフォーマンスを改善できる可能性があります。

バッファー・プール・データ・ページをディスクに書き込む理由は次のとおりです。

- バッファー・プール内のページを解放して、次のページを読み取れるようにする。
- バッファー・プールを空にする。

システムがあるページを書き込むときに、新しいページのためのスペースを用意するとは限りません。そのページが更新されていなければ、単純に置換されます。このエレメントでは、このような置換はカウントされません。

データ・ページは、バッファー・プール・スペースが必要になる前に、非同期ページ・クリーナー・エージェントによって書き込まれることがあります。それについては、**pool_async_data_writes** モニター・エレメントによって報告されます。これらの非同期ページ書き込みは、同期ページ書き込みと合わせて、このエレメントの値に含まれます。

このパーセンテージを計算するときは、バッファー・プールを最初に埋めるために必要となる物理読み取り数は無視してください。書き込みページ数は、次のように求めます。

1. アプリケーションを実行します (バッファーをロードするため)。
2. このエレメントの値を書き取ります。
3. アプリケーションを再び実行します。
4. このエレメントの新しい値からステップ 2 で記録した値を引きます。

アプリケーションを終了してから次に実行するまでのあいだにバッファー・プールの割り振りが解除されるのを防止するには、次のいずれかを行います。

- ACTIVATE DATABASE コマンドを使用してデータベースを活動化する。
- アイドル状態のアプリケーションをデータベースに接続する。

すべてのアプリケーションがデータベースを更新するような場合は、ほとんどのバッファー・プール・ページが更新されたデータを含んでおり、これをディスクに書き込む必要があるので、バッファー・プールのサイズを大きくしてもパフォーマンスはあまり改善されません。ただし、更新されたページを書き出す前に、ほかの作業単位がこのページを使用できる場合は、バッファー・プールが書き込み操作と読み取り操作を節約できるので、パフォーマンスが向上する場合があります。

pool_drty_pg_steal_clns - 起動されたバッファー・プール・ビクティム・ページ・クリーナー : モニター・エレメント

データベースのビクティム・バッファー置換の際に同期書き込みが必要になりページ・クリーナーが呼び出された回数。

表 805. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE

表 806. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 807. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法

次の数式を使用すると、クリーナー呼び出しの合計に占めるこのエレメントのパーセンテージを計算できます。

```
   pool_drty_pg_steal_clns
/ (pool_drty_pg_steal_clns
 + pool_drty_pg_thrsh_clns
 + pool_lsn_gap_clns)
```

この比率が低い場合は、定義したページ・クリーナー数が多すぎることを示します。 **chngpgs_thresh** 構成パラメーターをあまり低く設定すると、後にダーティー・ページになるページを書き出す可能性があります。クリーニングをあまり積極的にすると、バッファー・プールの 1 つの目的である、書き込みをできるだけ遅らせることができなくなります。

この比率が高い場合は、定義したページ・クリーナーの数が十分でないことを示している可能性があります。十分なページ・クリーナーがないと、障害時のリカバリー時間が長くなります。

DB2_USE_ALTERNATE_PAGE_CLEANING レジストリー変数が OFF の場合は、以下のようになります。

- **pool_drty_pg_steal_clns** モニター・エレメントがモニター・ストリーム中に挿入される。
- **pool_drty_pg_steal_clns** モニター・エレメントが、データベースのビクティム・バッファー置換の際に同期書き込みが必要になり、ページ・クリーナーが呼び出された回数をカウントする。

DB2_USE_ALTERNATE_PAGE_CLEANING レジストリー変数が ON の場合は、以下のようになります。

- **pool_drty_pg_steal_clns** モニター・エレメントが、モニター・ストリーム中に 0 を挿入する。
- ビクティム・バッファー置換の際に同期書き込みが必要でも、ページ・クリーナーは明示的に起動されない。データベースまたは特定のバッファー・プール用に構成されているページ・クリーナーの数が適切かどうかを判別するには、**pool_no_victim_buffer** モニター・エレメントを参照してください。

注: ダーティー・ページはディスクに書き出されますが、バッファー・プールに新しいページを読み取るためのスペースが必要な場合を除いて、このページはバッファー・プールからすぐには除去されません。

pool_drty_pg_thrsh_clns - 起動されたバッファー・プールしきい値クリーナー : モニター・エレメント

バッファー・プールがデータベースのダーティー・ページしきい値基準に達したためにページ・クリーナーが呼び出された回数。

表 808. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE

表 809. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 810. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 しきい値は **chngpgs_thresh** 構成パラメーターによって設定されます。この値は、バッファー・プール・サイズに適用されるパーセンテージです。プール内のダーティー・ページ数がこの値を超えると、クリーナーを起動します。

chngpgs_thresh 構成パラメーターの設定値が低すぎると、ページの書き出しが早すぎて、再読み取りが必要になる可能性があります。設定値が高すぎると、累積されるページ数が多くなり、ページを同期的に書き出す必要が生じる場合があります。

DB2_USE_ALTERNATE_PAGE_CLEANING レジストリー変数が OFF の場合は、以下のようになります。

- **pool_drty_pg_thrsh_clns** モニター・エレメントがモニター・ストリーム中に挿入される。
- **pool_drty_pg_thrsh_clns** モニター・エレメントが、バッファー・プールがデータベースのダーティー・ページしきい値基準に達したためにページ・クリーナーが呼び出された回数をカウントする。

DB2_USE_ALTERNATE_PAGE_CLEANING レジストリー変数が ON の場合は、以下のようになります。

- **pool_drty_pg_thrsh_clns** モニター・エレメントがモニター・ストリーム中に 0 を挿入する。
- ページ・クリーナーは、基準値によって起動されるのを待たず、常時アクティブで、使用可能なビクティム用の空きバッファーが十分あるか確認する。

pool_id メモリー・プール ID

メモリー・プールのタイプ。

表 811. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	memory_pool	基本
データベース	memory_pool	基本
アプリケーション	memory_pool	基本

表 812. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_dbmemuse	-
接続	event_connmemuse	-

使用法

システム・メモリーの使用量を追跡するときに、この値と **pool_max_size**、**pool_cur_size**、および **pool_watermark** を組み合わせて使用します。

pool_id を使用すると、システム・モニター出力に示されているメモリー・プールを識別できます。sqlmon.h に、さまざまなメモリー・プール ID があります。通常の操作条件では、次に示すプールの 1 つ以上が該当します。

API 定数	説明
SQLM_HEAP_APPLICATION	アプリケーション・ヒープ
SQLM_HEAP_DATABASE	データベース・ヒープ
SQLM_HEAP_LOCK_MGR	ロック・マネージャー・ヒープ
SQLM_HEAP_UTILITY	バックアップ/リストア/ユーティリティー・ヒープ
SQLM_HEAP_STATISTICS	統計ヒープ
SQLM_HEAP_PACKAGE_CACHE	パッケージ・キャッシュ・ヒープ
SQLM_HEAP_CAT_CACHE	カタログ・キャッシュ・ヒープ
SQLM_HEAP_MONITOR	データベース・モニター・ヒープ
SQLM_HEAP_STATEMENT	ステートメント・ヒープ
SQLM_HEAP_FCMBP	FCMBP ヒープ
SQLM_HEAP_IMPORT_POOL	インポート・プール
SQLM_HEAP_OTHER	その他のメモリー
SQLM_HEAP_BP	バッファー・プール・ヒープ
SQLM_HEAP_APPL_SHARED	アプリケーション共有ヒープ
SQLM_HEAP_SHARED_SORT	ソート共有ヒープ

pool_index_l_reads - バッファー・プール索引の論理読み取り : モニター・エレメント

REGULAR 表スペースおよび LARGE 表スペースのバッファー・プール (論理) から要求された、索引ページの数を示します。

表 813. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 813. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 814. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 815. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
表スペース	event_tablespace	-
接続	event_conn	-
ステートメント	event_stmt	-
アクティビティー	event_activity	バッファー・プール、ステートメント

使用法

このカウントには、次の索引ページへのアクセスが含まれます。

- データベース・マネージャーがページの処理を必要としたときにバッファー・プールにすでにあるデータ。
- データベース・マネージャーがページを処理する前にバッファー・プールに読み取られたデータ。

pool_index_p_reads と組み合わせて使用すると、次の数式を使用してバッファー・プールの索引ページ・ヒット率を計算できます。

```
1 - ((pool_index_p_reads - pool_async_index_reads) / pool_index_l_reads)
```

pool_data_l_reads および **pool_data_p_reads** モニター・エレメントを使用して、次の数式でバッファー・プールの全体的なデータ・ページ・ヒット率を計算します。

```
1 - ((pool_data_p_reads - pool_async_data_reads) / pool_data_l_reads)
```

ヒット率が低い場合は、バッファー・プール・ページ数を増やすと、パフォーマンスが向上する場合があります。

pool_index_p_reads - バッファー・プール索引の物理読み取り : モニター・エレメント

REGULAR 表スペースおよび LARGE 表スペースの表スペース・コンテナー (物理) から読み取られた、索引ページの数を示します。

表 816. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 816. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 817. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 818. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
表スペース	event_tablespace	-
接続	event_conn	-
ステートメント	event_stmt	-
アクティビティー	event_activity	バッファー・プール、ステートメント

使用法

pool_index_l_reads モニター・エレメントと組み合わせて使用すると、次の数式を使用してバッファー・プールの索引ページ・ヒット率を計算できます。

```
1 - ((pool_index_p_reads - pool_async_index_reads) / pool_index_l_reads)
```

pool_index_writes - バッファー・プール索引の書き込み : モニター・エレメント

バッファー・プール索引ページがディスクに物理的に書き込まれた回数。

表 819. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 820. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 821. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
表スペース	event_tablespace	-
接続	event_conn	-

使用法

データ・ページと同様に、バッファー・プール索引ページは以下の理由でディスクに書き込まれます。

- バッファー・プール内のページを解放して、次のページを読み取れるようにする。
- バッファー・プールを空にする。

システムがあるページを書き込むときに、新しいページのためのスペースを用意するとは限りません。そのページが更新されていなければ、単純に置換されます。このエレメントでは、このような置換はカウントされません。

索引ページは、バッファー・プール・スペースが必要になる前に、非同期ページ・クリーナー・エージェントにより書き込まれます。非同期索引ページの書き込みは、同期索引ページの書き込みと合わせて、このエレメントの値に含まれます (**pool_async_index_writes** モニター・エレメントを参照)。

pool_index_p_reads モニター・エレメントの値のパーセンテージが高いためにバッファー・プールの索引ページがディスクに書き込まれる場合は、データベースで利用可能なバッファー・プール・ページ数を増やすとパフォーマンスを改善できる可能性があります。

このパーセンテージを計算するときは、バッファー・プールを最初に埋めるために必要となる物理読み取り数は無視してください。書き込みページ数は、次のように求めます。

1. アプリケーションを実行します (バッファーをロードするため)。
2. このエレメントの値を書き取ります。
3. アプリケーションを再び実行します。
4. このエレメントの新しい値からステップ 2 で記録した値を引きます。

アプリケーションを終了してから次に実行するまでのあいだにバッファー・プールの割り振りが解除されるのを防止するには、次のいずれかを行います。

- ACTIVATE DATABASE コマンドを使用してデータベースを活動化する。
- アイドル状態のアプリケーションをデータベースに接続する。

すべてのアプリケーションがデータベースを更新するような場合は、ほとんどのページが更新されたデータを含んでおり、これをディスクに書き込む必要があるので、バッファー・プールのサイズを大きくしてもパフォーマンスはあまり改善されません。

pool_lsn_gap_clns - 起動されたバッファー・プール・ログ・スペース・クリーナー : モニター・エレメント

使用されているロギング・スペースがデータベースの定義済み基準に達したためにページ・クリーナーが呼び出された回数。

表 822. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE

表 823. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 824. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法

このエレメントは、ロギングに十分なスペースがあるかどうか、またログ・ファイルやさらに大きなログ・ファイルの追加が必要かどうかを判別するときに利用できます。

ページ・クリーニングの基準は、 **softmax** 構成パラメーターの設定値により決定されます。バッファー・プール内の最も古いページに含まれている更新内容が現行ログ位置と比較して基準値よりも古いログ・レコードにより記述されている場合に、ページ・クリーナーが起動される。

DB2_USE_ALTERNATE_PAGE_CLEANING レジストリー変数が OFF の場合は、以下のようになります。

- **pool_lsn_gap_clns** モニター・エレメントがモニター・ストリーム中に挿入される。
- バッファー・プール内の最も古いページに含まれている更新内容が現行ログ位置と比較して基準値よりも古いログ・レコードにより記述されている場合に、ページ・クリーナーが起動される。

DB2_USE_ALTERNATE_PAGE_CLEANING レジストリー変数が ON の場合は、以下のようになります。

- **pool_lsn_gap_clns** モニター・エレメントがモニター・ストリーム中に 0 を挿入する。
- ページ・クリーナーが、基準値によって起動されるのを待たずに、先行してページを書き込む。

pool_no_victim_buffer - バッファー・プールの非ビクティム・バッファー数 : モニター・エレメント

エージェントに、事前選択された使用可能なビクティム・バッファーがなかった回数。

表 825. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE

表 826. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 827. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
表スペース	event_tablespace	-

使用法 このエレメントは、ページ・クリーニングを先行して行う際に、特定のバッファー・プールに十分なページ・クリーナーがあるかどうかを判断するときに利用できます。

DB2_USE_ALTERNATE_PAGE_CLEANING レジストリー変数が ON の場合、pool_no_victim_buffer エレメントは、エージェントが即時使用に対応する事前選択されたビクティム・バッファーを見つけることができずに、バッファー・プールで適切なビクティム・バッファーを検索することを余儀なくされた回数をカウントします。

pool_no_victim_buffer エレメントの値が、バッファー・プールへの論理読み取りの数と比べて大きい場合、DB2 データベース・システムは、十分な数の適切なビクティムを確実に使用可能にしておくのが難しくなります。ページ・クリーナーの数を増やすと、DB2 が事前選択されたビクティム・バッファーを提供する能力も向上します。

DB2_USE_ALTERNATE_PAGE_CLEANING レジストリー変数が OFF の場合、pool_no_victim_buffer エレメントは予測値を持つわけではないので、無視しても問題ありません。この構成の場合、DB2 データベース・システムは、事前選択されたビクティム・バッファーをエージェントに対して使用可能にしておこうとしないため、バッファー・プールにアクセスするときには、大抵の場合、エージェントがバッファー・プールでビクティム・バッファーを検索する必要が生じます。

pool_read_time - バッファー・プール物理読み取り時間の合計 : モニター・エレメント

すべてのタイプの表スペースについて、表スペース・コンテナー (物理) からデータ・ページおよび索引ページを読み取るために費やされた合計時間を示します。この値はミリ秒単位で示されます。

表 828. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE

表 828. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 829. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 830. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
データベース	event_db	-
表スペース	event_tablespace	-
接続	event_conn	-

使用法

このエレメントと **pool_data_p_reads** および **pool_index_p_reads** モニター・エレメントを一緒に使用して、ページ読み取りの平均時間を計算します。この平均値は、入出力待ちがあるかどうかを示すので重要です。これにより、データをほかの装置に移動すべきかどうかがわかります。

データベースおよび表スペースのレベルでは、このエレメントには **pool_async_read_time** モニター・エレメントの値が含まれます。

pool_secondary_id メモリー・プール 2 次 ID

モニター・データを戻す対象となるメモリー・プールを判別するために役立つ追加 ID。

エレメント ID

pool_secondary_id

エレメント・タイプ

情報

表 831. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	memory_pool	基本
データベース	memory_pool	基本
アプリケーション	memory_pool	基本

表 832. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_dbmemuse	-
接続	event_connmemuse	-

使用法 モニター・データを戻す対象となるメモリー・プールを判別するために pool_id と共に使用します。 pool_secondary_id のデータは必要な場合のみ表示されます。例えば、示された pool_id が、モニター・データがどのバッファー・プールに関連するかを判別するためのバッファー・プール・ヒープである場合に pool_secondary_id のデータは表示されます。

データベースの作成時に、(IBMDEFAULTBP という) デフォルトのバッファー・プールがデータベースに作成され、そのサイズはプラットフォームによって決まります。このバッファー・プールは「1」という 2 次 ID を持ちます。このバッファー・プール、およびユーザーが作成するすべてのバッファー・プールに加えて、それぞれ異なるページ・サイズに対応するいくつかのシステム・バッファー・プールがデフォルトで作成されます。これらのバッファー・プールの ID は、pool_secondary_id のスナップショットに次のように表示される可能性があります。

- System 32k buffer pool
- System 16k buffer pool
- System 8k buffer pool

- System 4k buffer pool

pool_temp_data_l_reads - バッファー・プール一時データの論理読み取り : モニター・エレメント

TEMPORARY 表スペースのバッファー・プール (論理) から要求された、データ・ページの数を示します。

表 833. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 834. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール

表 834. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 835. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
データベース	event_db	-
表スペース	event_tablespace	-
接続	event_conn	-
ステートメント	event_stmt	-
アクティビティー	event_activity	バッファー・プール、ステートメント

使用法

pool_temp_data_p_reads エレメントと組み合わせて使用すると、次の数式で TEMPORARY 表スペースにあるバッファー・プールのデータ・ページ・ヒット率を計算できます。

```
1 - (pool_temp_data_p_reads / pool_temp_data_l_reads)
```

全体のバッファー・プール・ヒット率は、次のように計算できます。

```
    1 - ((pool_data_p_reads + pool_xda_p_reads +
pool_index_p_reads + pool_temp_data_p_reads
+ pool_temp_xda_p_reads + pool_temp_index_p_reads )
     / (pool_data_l_reads + pool_xda_l_reads + pool_index_l_reads +
     pool_temp_data_l_reads + pool_temp_xda_l_reads
     + pool_temp_index_l_reads )) * 100%
```

この計算には、バッファー・プールによってキャッシュされているすべてのページ (索引とデータ) が含まれます。

pool_temp_data_p_reads - バッファー・プール一時データの物理読み取り : モニター・エレメント

TEMPORARY 表スペースの表スペース・コンテナー (物理) から読み取られた、データ・ページの数を示します。

表 836. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 837. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

表 837. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 838. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
データベース	event_db	-
表スペース	event_tablespace	-
接続	event_conn	-
ステートメント	event_stmt	-
アクティビティー	event_activity	バッファー・プール、ステートメント

使用法

ステートメント・レベルでバッファー・プール情報を記録する機能は、 API および CLP スナップショット要求用にサポートされています。

pool_temp_data_l_reads エレメントと組み合わせて使用すると、次の数式で TEMPORARY 表スペースにあるバッファー・プールのデータ・ページ・ヒット率を計算できます。

```
1 - (pool_temp_data_p_reads / pool_temp_data_l_reads)
```

pool_temp_index_l_reads - バッファー・プール一時索引の論理読み取り : モニター・エレメント

TEMPORARY 表スペースのバッファー・プール (論理) から要求された、索引ページの数を示します。

表 839. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE

表 839. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 840. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 841. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
データベース	event_db	-
表スペース	event_tablespace	-
接続	event_conn	-
ステートメント	event_stmt	-
アクティビティー	event_activity	バッファー・プール、ステートメント

使用法

このエレメントを **pool_temp_index_p_reads** エレメントと組み合わせて使用すると、次の数式を使用して TEMPORARY 表スペースにあるバッファー・プールの索引ページ・ヒット率を計算できます。

```
1 - (pool_temp_index_p_reads / pool_temp_index_l_reads)
```

pool_temp_index_p_reads - バッファー・プール一時索引の物理読み取り : モニター・エレメント

TEMPORARY 表スペースの表スペース・コンテナー (物理) から読み取られた、索引ページの数を示します。

表 842. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE

表 842. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 843. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 844. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
データベース	event_db	-
表スペース	event_tablespace	-
接続	event_conn	-
ステートメント	event_stmt	-

表 844. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	バッファー・プール、ステートメント

使用法

このエレメントを **pool_temp_index_l_reads** エレメントと組み合わせて使用すると、次の数式を使用して TEMPORARY 表スペースにあるバッファー・プールの索引ページ・ヒット率を計算できます。

```
1 - (pool_temp_index_p_reads / pool_temp_index_l_reads)
```

pool_temp_xda_l_reads - バッファー・プール一時 XDA データの論理読み取り : モニター・エレメント

TEMPORARY 表スペースのバッファー・プール (論理) から要求された、XML ストレージ・オブジェクト (XDA) データのページの数を示します。

表 845. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 845. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 846. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 847. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
データベース	event_db	-
表スペース	event_tablespace	-
接続	event_conn	-
ステートメント	event_stmt	-
アクティビティー	event_activity	バッファー・プール、ステートメント

使用法

pool_temp_xda_l_reads モニター・エレメントを **pool_temp_xda_p_reads**、**pool_temp_data_l_reads**、および **pool_temp_data_p_reads** モニター・エレメントと組み合わせて使用すると、TEMPORARY 表スペースにあるバッファー・プールのデータ・ページ・ヒット率を次の数式で計算できます。

```
1 - ((pool_temp_data_p_reads + pool_temp_xda_p_reads)
      / (pool_temp_data_l_reads + pool_temp_xda_l_reads))
```

pool_temp_xda_p_reads - バッファー・プール一時 XDA データの物理読み取り : モニター・エレメント

TEMPORARY 表スペースの表スペース・コンテナー (物理) から読み取られた、XML ストレージ・オブジェクト (XDA) データのページの数を示します。

表 848. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 849. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール

表 849. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 850. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
データベース	event_db	-
表スペース	event_tablespace	-
接続	event_conn	-
ステートメント	event_stmt	-
アクティビティー	event_activity	バッファー・プール、ステートメント

使用法

pool_temp_xda_p_reads モニター・エレメントを **pool_temp_xda_l_reads**、**pool_temp_data_l_reads**、および **pool_temp_data_p_reads** モニター・エレメントと組み合わせて使用すると、TEMPORARY 表スペースにあるバッファー・プールのデータ・ページ・ヒット率を次の数式で計算できます。

```
1 - ((pool_temp_data_p_reads + pool_temp_xda_p_reads) /
(pool_temp_data_l_reads + pool_temp_xda_l_reads))
```

pool_watermark メモリー・プール水準点

メモリー・プール作成後のその最大サイズ。

エレメント ID

pool_watermark

エレメント・タイプ

情報

表 851. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	memory_pool	基本
データベース	memory_pool	基本

表 851. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	memory_pool	基本

表 852. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_dbmemuse	-
接続	event_connmemuse	-

使用法 継続的に稼働しているシステムの場合は、 *pool_watermark* と *pool_config_size* のエレメントを組み合わせて使用すると、メモリーに関する潜在的な問題を予測できます。

例えば、一定間隔 (例えば 1 日に 1 回) でスナップショットを取り、*pool_watermark* と *pool_config_size* の値を調べます。 *pool_watermark* の値が *pool_config_size* の値に近づく場合は (メモリー関連のトラブルが起こる可能性を示しています)、メモリー・プールのサイズを大きくする必要があることを示します。

pool_write_time - バッファー・プール物理書き込み時間の合計 : モニター・エレメント

データ・ページまたは索引ページをバッファー・プールからディスクに物理的に書き込むのに要した合計時間を示します。経過時間はミリ秒単位で示されます。

表 853. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 853. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 854. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 855. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
表スペース	event_tablespace	-
接続	event_conn	-

使用法

このエレメントを **buffer_pool_data_writes** および **pool_index_writes** モニター・エレメントと一緒に使用して、ページ書き込みの平均時間を計算します。この平均値は、入出力待ちがあるかどうかを示すので重要です。これにより、データをほかの装置に移動すべきかどうかがわかります。

データベースおよび表スペースのレベルでは、このエレメントには **pool_async_write_time** モニター・エレメントの値が含まれます。

pool_xda_l_reads - バッファー・プール XDA データの論理読み取り : モニター・エレメント

REGULAR 表スペースおよび LARGE 表スペースのバッファー・プール (論理) から要求された、XML ストレージ・オブジェクト (XDA) のデータ・ページの数を示します。

表 856. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 857. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール

表 857. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 858. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
表スペース	event_tablespace	-
接続	event_conn	-
ステートメント	event_stmt	-
アクティビティー	event_activity	バッファー・プール、ステートメント

使用法

このカウントには、次のデータへのアクセスが含まれます。

- データベース・マネージャーがページの処理を必要としたときにバッファー・プールにすでにあるデータ。
- データベース・マネージャーがページを処理する前にバッファー・プールに読み取られたデータ。

pool_xda_l_reads、**pool_xda_p_reads**、**pool_data_l_reads**、および **pool_data_p_reads** モニター・エレメントを使用して、次の数式でバッファー・プールのデータ・ページ・ヒット率を計算します。

```
1 - ((pool_data_p_reads + pool_xda_p_reads)
   / (pool_data_l_reads + pool_xda_l_reads))
```

全体のバッファー・プール・ヒット率は、次のように計算できます。

```
    1 - ((pool_data_p_reads + pool_xda_p_reads +
pool_index_p_reads + pool_temp_data_p_reads
+ pool_temp_xda_p_reads + pool_temp_index_p_reads )
     / (pool_data_l_reads + pool_xda_l_reads + pool_index_l_reads +
     pool_temp_data_l_reads + pool_temp_xda_l_reads
     + pool_temp_index_l_reads )) * 100%
```

この計算には、バッファー・プールによってキャッシュされているすべてのページ (索引とデータ) が含まれます。

バッファー・プール・サイズを大きくすると、一般的にヒット率は高くなりますが、ある点を超えると逆に低くなります。理想的には、データベース全体を保管できるような大きなバッファー・プールを割り振ることができれば、システムが稼働中のヒット率は 100% になります。しかし、現実的にはそうしたことは起こりません。使用するデータのサイズとそのデータへのアクセス方法によってヒット率の重要度は異なります。非常に大きなデータベースでアクセスが均等な場合は、ヒット率が低くなります。表が非常に大きな場合は、対応する方法はほとんどありません。このような場合、より小さく頻繁にアクセスがある表、および索引に着目するのが賢明です。

pool_xda_p_reads - バッファー・プール XDA データの物理読み取り : モニター・エレメント

REGULAR 表スペースおよび LARGE 表スペースの表スペース・コンテナー (物理) から読み取られた、XML ストレージ・オブジェクト (XDA) のデータ・ページの数を示します。

表 859. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 859. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 860. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール
アプリケーション	stmt	バッファー・プール
動的 SQL	dynsql	バッファー・プール、ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 861. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
表スペース	event_tablespace	-
接続	event_conn	-
ステートメント	event_stmt	-
アクティビティー	event_activity	バッファー・プール、ステートメント

使用法

pool_async_xda_reads および **pool_xda_p_reads** モニター・エレメントを使用して、XML ストレージ・オブジェクト・データ・ページ上で同期的に実行された物理読み取りの数 (つまり、データベース・マネージャーのエージェントが XML データに対して実行したデータ・ページの物理読み取り数) を計算します。次の数式を使用します。

```
pool_xda_p_reads - pool_async_xda_reads
```

非同期読み取り数と同期読み取り数を比較すると、プリフェッチャーの動作状態がわかります。このエレメントは、**num_ioservers** 構成パラメーターを調整するときに役に立ちます。

pool_xda_l_reads、**pool_xda_p_reads**、**pool_data_l_reads**、および **pool_data_p_reads** モニター・エレメントを使用して、次の数式でバッファー・プールのデータ・ページ・ヒット率を計算します。

```
1 - ((pool_data_p_reads + pool_xda_p_reads)
   / (pool_data_l_reads + pool_xda_l_reads))
```

pool_xda_writes - バッファー・プール XDA データの書き込み : モニター・エレメント

XML ストレージ・オブジェクト (XDA) のバッファー・プール・データ・ページがディスクに物理的に書き込まれた回数を示します。

表 862. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 862. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 863. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 864. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
表スペース	event_tablespace	-
接続	event_conn	-

使用法

このモニター・エレメントは、データベースで使用可能なバッファー・プール・ページの数を増やすことによりパフォーマンスが向上するかを評価する助けとなります。 XML データを含むデータベースの場合、バッファー・プール・ページの読み取りに対する書き込みの比率を、XML データおよびリレーショナル・データの両方のタイプについて考慮する必要があります。XML データの場合は **pool_xda_writes**

および **pool_xda_p_reads** モニター・エレメントを使用し、リレーショナル・データの場合は **pool_data_writes** および **pool_data_p_reads** モニター・エレメントを使用します。

pool_xda_l_reads、**pool_xda_p_reads**、**pool_data_l_reads**、および **pool_data_p_reads** モニター・エレメントを使用して、次の数式でバッファー・プールのデータ・ページ・ヒット率を計算します。

```
1 - ((pool_data_p_reads + pool_xda_p_reads)
   / (pool_data_l_reads + pool_xda_l_reads))
```

post_shrthreshold_hash_joins ポストしきい値ハッシュ結合

ソート・メモリー・スロットル・アルゴリズムによってスロットルして戻されたハッシュ結合の合計数。スロットルされたハッシュ結合は、ソート・メモリー・マネージャーが要求するメモリーよりも少ないメモリーが付与されたハッシュ結合です。

表 865. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	-

スナップショット・モニターの場合、このカウンターはリセットできます。

表 866. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

共有ソート・ヒープから割り振られるメモリーがデータベース構成パラメーター *sheapthres_shr* により設定された制限に近づくと、ハッシュ結合はスロットルして戻されます。システムが適切に構成されていない場合、このスロットルによって、*sheapthres_shr* 制限を超えたオーバーフロー数が大幅に削減されます。このエレメントで報告されるデータは、共有ソート・ヒープから割り振られるメモリーを使用したハッシュ結合のみを反映します。

post_shrthreshold_sorts - ポスト共有しきい値ソート : モニター・エレメント

ソート・メモリー・スロットル・アルゴリズムによってスロットルして戻されたソートの合計数。スロットルされたソートは、ソート・メモリー・マネージャーが要求するメモリーよりも少ないメモリーが付与されたソートです。ソートに対するメモリーの割り振りがデータベース構成パラメーター *sheapthres_shr* により設定された制限に近づくと、ソートはスロットルして戻されます。システムが適切に構成されていない場合、このスロットルによって、*sheapthres_shr* 制限を超えたオーバーフロー数が大幅に削減されます。このエレメントで報告されるデータは、共有ソート・ヒープから割り振られるメモリーを使用したソートのみを反映します。

表 867. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 868. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	Sort

スナップショット・モニターの場合、このカウンターはリセットできます。

表 869. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

表 869. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

post_threshold_hash_joins ハッシュ結合のしきい値

共有または専用のソート・ヒープ・スペースが同時使用されていたためにハッシュ結合ヒープ要求が制限された合計回数。

エレメント ID

post_threshold_hash_joins

エレメント・タイプ

カウンター

表 870. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 この値が大きい (hash_join_overflows の 5% より大きい) 場合は、ソート・ヒープのしきい値を大きくしてください。

post_threshold_olap_funcs OLAP 関数のしきい値 : モニター・エレメント

ソート・ヒープしきい値を超えた後にソート・ヒープを要求した OLAP 関数の数。

表 871. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

ソート、ハッシュ結合、および OLAP 関数は、ソート・ヒープを使用する操作の例です。通常の条件では、データベース・マネージャーは、sortheap 構成パラメーターによって指定された値を使用して、ソート・ヒープを割り振ります。ソート・ヒープに割り振られたメモリー量がソート・ヒープのしきい値を超えると (sheapthres 構成パラメーター)、データベース・マネージャーは、sortheap 構成パラメーターが指定する値よりも低い値を使用して以降のソート・ヒープを割り振ります。

ソート・ヒープのしきい値に達すると、その後に開始した OLAP 関数では実行するための十分なメモリー量を得られないことがあります。

ソート、ハッシュ結合、OLAP 関数のパフォーマンス、およびシステム全体のパフォーマンスを改善するには、ソート・ヒープしきい値およびソート・ヒープ・サイズの構成パラメーターを変更します。

このエレメントの値が高い場合は、ソート・ヒープしきい値 (sheapthres) を上げます。

post_threshold_sorts - ポストしきい値ソート : モニター・エレメント

ソート・ヒープしきい値に達した後でヒープを要求したソートの数。

表 872. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 873. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	Sort

スナップショット・モニターの場合、このカウンターはリセットできます。

表 874. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

使用法

通常の条件では、データベース・マネージャーは、 **sortheap** 構成パラメーターによって指定された値を使用して、ソート・ヒープを割り振ります。ソート・ヒープに割り振られたメモリー量がソート・ヒープのしきい値を超えると (**sheapthres** 構成パラメーター)、データベース・マネージャーは、 **sortheap** 構成パラメーターが指定する値よりも低い値を使用してソート・ヒープを割り振ります。

システム上のアクティブ・ソートには、それぞれメモリーが割り振られるので、利用可能なシステム・メモリーからソートのためのメモリー量を多く取り過ぎることがあります。ソート・ヒープのしきい値を超えると、その後に開始したソートでは実行するための十分なメモリー量を得られないことがありますが、システム全体としてはそのほうが利点があります。ソート・ヒープしきい値およびソート・ヒープ・サイズの構成パラメーターを変更すると、ソート操作のパフォーマンスとシステム全体のパフォーマンスを改善できます。エレメントの値が高い場合は、次のいずれかを行うことができます。

- ソート・ヒープしきい値 (**sheapthres**) を上げる。
- SQL 照会で使用するソート数を少なくするか、小さくなるようにアプリケーションを調整する。

prefetch_wait_time - プリフェッチ待ち時間 : モニター・エレメント

アプリケーションが入出力サーバー (プリフェッチャー) によるバッファー・プールへのページのロードの終了を待機していた時間。値はミリ秒単位で示されます。

表 875. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
アプリケーション	appl	バッファー・プール

表 876. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 このエレメントを使用すると、入出力サーバーの数と入出力サーバーのサイズの変更を試すことができます。

prep_time 準備時間 : モニター・エレメント

アクティビティーが SQL ステートメントである場合に SQL ステートメントを準備するために必要な時間 (ミリ秒単位)。

表 877. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 878. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

使用法

このエレメントを使用すると、これが SQL アクティビティーであった場合、アクティビティーの合計存続時間のうち、どれだけの時間が SQL ステートメントを準備するために費やされたかを識別できます。

prep_time_best ステートメント最短準備時間 : モニター・エレメント

特定の SQL ステートメントの準備に要した最短時間 (ミリ秒単位)。

表 879. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
動的 SQL	dynsql	基本

使用法

この値を **prep_time_worst** とともに使用して、コンパイルに長い時間がかかる SQL ステートメントを識別します。

prep_time_worst ステートメント最長準備時間 : モニター・エレメント

特定の SQL ステートメントの準備に要した最長時間 (ミリ秒単位)。

表 880. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
動的 SQL	dynsql	基本

使用法

この値を **prep_time_best** とともに使用して、コンパイルに長い時間がかかる SQL ステートメントを識別します。

prev_uow_stop_time 直前の作業単位完了タイム・スタンプ

作業単位が完了した時刻です。

エレメント ID

prev_uow_stop_time

エレメント・タイプ

timestamp

表 881. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	作業単位、タイム・スタンプ
DCS アプリケーション	dcs_appl	作業単位、タイム・スタンプ

使用法 このエレメントと *uow_stop_time* を組み合わせて使用すると、 COMMIT と ROLLBACK のポイント間の合計経過時間を計算できます。 *uow_start_time* と組み合わせて使用すると、作業単位間のアプリケーションの合計時間を計算できます。次のいずれかの時刻を示します。

- アプリケーションが作業単位中の場合は、最後に作業単位が完了した時刻を示す。
- アプリケーションが作業単位中ではない場合は (アプリケーションが 1 つの作業単位を完了し、次の作業単位をまだ開始していない場合)、最後に完了した作業単位の直前の作業単位の停止時刻を示す。最後に完了した作業単位の停止時刻は、uow_stop_time で示す。
- アプリケーションが最初の作業単位中の場合は、データベース接続要求完了時刻となる。

priv_workspace_num_overflows 専用ワークスペースのオーバーフロー回数

割り振られたメモリーの境界から専用ワークスペースがオーバーフローした回数。

注: このモニター・エレメントは、使用しないでください。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 882. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 883. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 このエレメントと priv_workspace_size_top を組み合わせて使用すると、オーバーフローを防止するのに専用ワークスペースのサイズを大きくする必要があるかどうかを判別できます。専用ワークスペースがオーバーフローすると、パフォーマンスが低下するだけではなく、エージェントの専用メモリーから割り振られたほかのヒープでメモリー不足エラーが発生することがあります。

データベース・レベルでは、「専用ワークスペースの最大サイズ」のある専用ワークスペースとして報告された専用ワークスペースがこのエレメントの報告の対象となります。アプリケーション・レベルでは、現行アプリケーションにサービスを提供した各エージェントのワークスペースがオーバーフローした回数となります。

priv_workspace_section_inserts 専用ワークスペース・セクション挿入

専用ワークスペースへの、アプリケーションによる SQL セクション挿入数。

注: このモニター・エレメントは、使用しないでください。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 884. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 885. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 実行可能セクションの作業用コピーは、専用ワークスペース内に保管されます。

このカウンターは、コピーが使用できなかったために挿入が必要だった場合を示します。データベース・レベルでは、データベース内のすべての専用ワークスペースを対象に、すべてアプリケーションでの累計挿入数を示します。アプリケーション・レベルでは、このアプリケーションの専用ワークスペース内にあるすべてのセクションを対象とした累計挿入数を示します。

エージェントが異なるアプリケーションに関連付けられているようなコンセントレーター環境では、新しいエージェントに必要な使用可能なセクションが専用ワークスペース内にない場合に、専用ワークスペースの追加挿入が必要になります。

priv_workspace_section_lookups 専用ワークスペース・セクション検索

エージェントの専用ワークスペースでの、アプリケーションによる SQL セクション検索数。

注: このモニター・エレメントは、使用しないでください。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 886. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 887. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 各アプリケーションは、自分に代わって作業するエージェントの専用ワークスペースにアクセスできます。

このカウンターは、アプリケーション用の特定セクションを見つけるために専用ワークスペースがアクセスされた回数を示します。データベース・レベルでは、データベース内のすべての専用ワークスペースを対象に、すべてのアプリケーションでの累計検索数を示します。アプリケーション・レベルでは、このアプリケーションの専用ワークスペース内にあるすべてのセクションを対象とした累計検索数を示します。

このエレメントと「専用ワークスペース・セクション挿入」を組み合わせて使用すると、専用ワークスペースのサイズを調整できます。専用ワークスペースのサイズをコントロールしているのは、applheapsz 構成パラメーターです。

priv_workspace_size_top 専用ワークスペースの最大サイズ

専用ワークスペースが到達した最大サイズ。

注: このモニター・エレメントは、使用しないでください。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 888. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

表 889. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 各エージェントには 1 つの専用ワークスペースがあり、エージェントがサービスを提供するアプリケーションはこれにアクセスをします。このエレメントは、アプリケーションにサービスを提供するエージェントが必要とする専用ワークスペースの最大バイト数を示します。データベース・レベルでは、現行データベースにアタッチされているすべてのエージェントが必要とする、すべての専用ワークスペースの最大バイト数を示します。アプリケーション・レベルでは、現行アプリケーションにサービスを提供したエージェントのすべての専用ワークスペースの中での最大サイズを示します。

専用ワークスペースがオーバーフローすると、エージェント専用メモリーにあるほかのエンティティーからメモリーを一時的に借用します。この結果、これらのエンティティーでメモリー不足エラーが発生したり、パフォーマンスが低下することがあります。 APPLHEAPSZ を大きくすると、オーバーフローの確率を低くすることができます。

product_name 製品名

実行中の DB2 インスタンスのバージョンの詳細情報。

表 890. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

progress_completed_units 完了した進行作業単位

現行フェーズの、完了した作業単位の数。

表 891. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	progress	基本

通常このエレメントの値は、ユーティリティーが作動するにつれて大きくなります。このエレメントは、常に *progress_total_units* 以下になります (両方のエレメントとも定義されている場合)。

注:

1. このエレメントが組み込まれていないユーティリティーが存在する可能性があります。
2. このエレメントは、 *progress_work_metric* モニター・エレメントで表示される単位で表されます。

使用法 このエレメントを使用して、フェーズ中の完了した作業の量を判別できます。このエレメントを単独で使用すると、実行中のユーティリティーのアクティビティーをモニターできます。このエレメントは、ユーティリティーの実行につれて連続的に大きくなるはずです。 *progress_completed_units* が長期間大きくならない場合は、ユーティリティーが停止している可能性があります。

progress_total_units を定義している場合は、このエレメントを使用して、完了した作業のパーセンテージを計算できます。

```
完了した作業のパーセンテージ = progress_completed_units /
progress_total_units* 100
```

progress_description 進行の記述

作業のフェーズの説明。

表 892. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	progress	基本

ロード・ユーティリティーの値の例には、以下が含まれます。

- DELETE
- LOAD
- REDO

使用法 このエレメントを使用して、フェーズの一般説明を取得します。

progress_list_attr 現在の進行リストの属性

このエレメントは、進行エレメントのリストを解釈する方法を記述します。

表 893. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	進行リスト	基本

使用法

このエレメントの値は、以下のいずれかの定数です。

- SQLM_ELM_PROGRESS_LIST_ATTR_SERIAL - リスト内のエレメントは順次フェーズのセットとして解釈されます。つまり、完了した処理数は、エレメント *n*+1 の完了した処理が最初に更新される前にエレメント *n* の合計処理数と等しくならなければなりません。この属性は、次のフェーズが開始する前に 1 つのフェーズが完全に完了する必要がある順次フェーズのセットで構成されるタスクの進行を記述するために使用されます。

- SQLM_ELM_PROGRESS_LIST_ATTR_CONCURRENT - 進行リスト内の任意のエレメントは、いつでも更新できます。

このエレメントを使用すると、進行リストのエレメントを更新する方法を判別できます。

progress_list_cur_seq_num 現行の進行リストのシーケンス番号

ユーティリティーに複数の連続したフェーズが含まれている場合、このエレメントは現行フェーズの番号を表示します。

表 894. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	progress_list	基本

使用法 このエレメントを使用して、ユーティリティーの連続したフェーズのうちの現在のフェーズを判別できます。 『progress_seq_num 進行シーケンス番号』を参照してください。

progress_seq_num 進行シーケンス番号

フェーズ番号。

注: 複数の実行フェーズから成るユーティリティーの場合のみ、フェーズ番号が表示されます。

表 895. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	progress	基本

使用法 このエレメントを使用して、複数フェーズ・ユーティリティー中のフェーズの順序を判別できます。このユーティリティーは、進行シーケンス番号の昇順でフェーズを実行します。複数フェーズ・ユーティリティーの現行フェーズを見つけるには、 *progress_seq_num* と、 *progress_list_current_seq_num* の値を突き合わせます。

progress_start_time 進行開始時刻

フェーズの開始を示すタイム・スタンプ。

表 896. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	progress	基本

使用法 このエレメントを使用すると、フェーズが開始した時点を判別できます。フェーズがまだ開始されていない場合は、このエレメントは省略されます。

progress_total_units 合計進行作業単位

フェーズを完了するために実行する作業の合計量。

表 897. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	progress	基本

合計作業量を定量化できないので、このエレメントが継続的に更新されるユーティリティーもあれば、合計作業を見積もれないので、このエレメントが完全に省略されるユーティリティーもあります。

このエレメントは、 *progress_work_metric* モニター・エレメントで表示される単位で表されます。

使用法 このエレメントを使用して、フェーズ中の作業の合計量を判別します。フェーズ中の完了した作業のパーセンテージを計算するには、このエレメントと *progress_completed_units* を併用してください。

```
完了した作業のパーセンテージ = progress_completed_units /
progress_total_units* 100
```

progress_work_metric 進行作業メトリック

progress_total_units エレメントと *progress_completed_units* エレメントを解釈するメトリック。

表 898. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	progress	基本

値の例には、以下のものがあります。

- SQLM_WORK_METRIC_BYTES
- SQLM_WORK_METRIC_EXTENTS

注:

1. このエレメントが組み込まれていないユーティリティーが存在する可能性があります。
2. このエレメントの値は sqlmon.h 中にあります。

使用法 このエレメントを使用して、 *progress_total_units* と *progress_completed_units* が報告メトリックとして使用しているものを判別します。

pseudo_deletes - 疑似削除 : モニター・エレメント

疑似空ページのすべてのキーは疑似削除されています。このモニター・エレメントは、削除された疑似空ページの数をレポートします。

表 899. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

pseudo_empty_pages - 疑似空ページ : モニター・エレメント

疑似空ページのすべてのキーは疑似削除されています。このモニター・エレメントは、疑似空として識別されたページの数をレポートします。

表 900. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

使用法

注: このモニター・エレメントは、疑似空ページの現在の数をレポートしません。

qp_query_id - Query patroller 照会 ID : モニター・エレメント

このアクティビティーが照会の場合に、Query Patroller によってそのアクティビティーに割り当てられた照会 ID。照会 ID が 0 の場合は、Query Patroller がこのアクティビティーに対して照会 ID を割り当てなかったことを意味します。

重要: qp_query_id モニター・エレメントは、Query Patroller 機能と関連付けられているため、推奨されません。 DB2 バージョン 9.5 で新しいワークロード管理フィーチャーが導入されたことで、バージョン 9.7 では Query Patroller とその関連コンポーネントは推奨されなくなりました。今後のリリースでは除去される可能性があります。

表 901. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

query_card_estimate 照会行数の見積もり

照会によって戻される行数の見積もり。

表 902. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント
DCS ステートメント	dcs_stmt	ステートメント

表 902. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

使用法 SQL コンパイラーによるこの見積もりは、ランタイムの実際のものと比較できます。

このエレメントは、 DB2 Connect をモニター中は次の SQL ステートメントに関する情報も戻します。

- INSERT、UPDATE、および DELETE

  影響を受ける行数を示します。

- PREPARE

  戻される行数の見積もり。 DRDA サーバーが DB2 Database for Linux, UNIX, and Windows、 DB2 for VM/VSE、または DB2 for OS/400® の場合にのみ収集します。

- FETCH

  フェッチされた行数に設定されます。 DRDA サーバーが DB2 for OS/400 の場合にのみ収集します。

DRDA サーバーで情報が収集されないと、エレメントはゼロに設定されます。

query_cost_estimate - 照会コストの見積もり : モニター・エレメント

SQL コンパイラーによって判別された照会の見積もりコスト。この値は timerons 単位で報告されます。

表 903. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

表 904. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント
DCS ステートメント	dcs_stmt	ステートメント
アクティビティー	event_activity	-

使用法 これにより、ランタイムの値とコンパイル時の見積もりの相関をとることができます。

このエレメントは、 DB2 Connect をモニター中は次の SQL ステートメントに関する情報も戻します。

- PREPARE

準備済み SQL ステートメントの相対コストを示します。

- FETCH

  取り出した行の長さが含まれています。 DRDA サーバーが DB2 for OS/400 の場合にのみ収集します。

DRDA サーバーで情報が収集されないと、エレメントはゼロに設定されます。

注: DRDA サーバーが DB2 for OS/390® and z/OS の場合は、この見積もりが 2**32 - 1 (符号なしロング変数を使用して表現できる最大整数) よりも大きい値になることがあります。この場合は、モニターがこのエレメントに戻す値は 2**32 - 1 になります。

queue_assignments_total キュー割り当ての合計 : モニター・エレメント

最後にリセットされてからこのしきい値キューに接続またはアクティビティーが割り当てられた回数。

表 905. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	-

使用法

このエレメントを使用すると、統計収集間隔により決定される特定の期間にアクティビティーまたは接続がこの特定のキューに入れられた回数を判別できます。 これは、キューのしきい値の効果性を判別する助けになります。

queue_size_top キュー・サイズの最上位 : モニター・エレメント

最後にリセットしてから到達したキュー・サイズの最大値。

表 906. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	-

使用法

このエレメントを使用すると、キューのしきい値の効果性を測定したり、キューイングが大きすぎる時を検出したりすることができます。

queue_time_total キュー時間の合計 : モニター・エレメント

最後にリセットされてからキューに置かれたすべての接続またはアクティビティーについて、このキューで費やされた合計時間。単位はミリ秒です。

表 907. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	-

使用法

このエレメントを使用すると、キューのしきい値の効果性を測定したり、キューイングが大きすぎる時を検出したりすることができます。

quiescer_agent_id 静止プログラム・エージェント ID

静止状態を保持するエージェントのエージェント ID。

エレメント ID
quiescer_agent_id

エレメント・タイプ
情報

表 908. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_quiescer	基本

使用法 このエレメントと quiescer_auth_id を組み合わせて使用すると、表スペースを静止させたユーザーを判別できます。

quiescer_auth_id 静止プログラム・ユーザー許可 ID

静止状態を保持するユーザーの許可 ID。

エレメント ID
quiescer_auth_id

エレメント・タイプ
情報

表 909. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_quiescer	基本

使用法 このエレメントを使用すると、表スペースを静止させたユーザーを判別できます。

quiescer_obj_id 静止プログラム・オブジェクト ID

表スペースを静止させたオブジェクトのオブジェクト ID。

エレメント ID
quiescer_obj_id

エレメント・タイプ
情報

表 910. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_quiescer	基本

使用法 このエレメントと quiescer_ts_id および quiescer_auth_id を組み合わせて使用すると、表スペースを静止させたオブジェクトを判別できます。このエレメントの値は、SYSCAT.TABLES ビューの TABLEID 列の値と一致します。

quiescer_state 静止プログラムの状態

行われている静止タイプ (例えば、「共有」、「更新する」、または「排他」)。

エレメント ID

quiescer_state

エレメント・タイプ

情報

表 911. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_quiescer	基本

使用法 このエレメントの値は、sqlutil.h の SQLB_QUIESCED_SHARE、SQLB_QUIESCED_UPDATE、または SQLB_QUIESCED_EXCLUSIVE の定数の値と一致します。

quiescer_ts_id 静止プログラム表スペース ID

表スペースを静止させたオブジェクトの表スペース ID。

エレメント ID

quiescer_ts_id

エレメント・タイプ

情報

表 912. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_quiescer	基本

使用法 このエレメントと quiescer_obj_id および quiescer_auth_id を組み合わせて使用すると、表スペースを静止させたユーザーを判別できます。このエレメントの値は、SYSCAT.TABLES ビューの TBSPACEID 列の値と一致します。

range_adjustment 範囲調整

この値は、1 つの範囲が実際に開始するコンテナー配列のオフセットを示します。

表 913. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_range	基本

使用法 このエレメントは、DMS 表スペースにのみ適用できます。

range_container_id 範囲コンテナー

範囲内でコンテナーを一意的に定義する整数。

表 914. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_range	基本

使用法 このエレメントは、DMS 表スペースにのみ適用できます。

range_end_stripe 終了ストライプ

この値は、1 つの範囲内の最後のストライプの番号を示します。

表 915. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_range	基本

使用法 このエレメントは、DMS 表スペースにのみ適用できます。

range_max_extent 範囲内の最大エクステント

この値は、1 つの範囲がマップする最大エクステント番号を示します。

表 916. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_range	基本

使用法 このエレメントは、DMS 表スペースにのみ適用できます。

range_max_page_number 範囲内の最大ページ

この値は、1 つの範囲がマップする最大ページ番号を示します。

表 917. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_range	基本

使用法 このエレメントは、DMS 表スペースにのみ適用できます。

range_num_containers 範囲内コンテナーの数

この値は、現行範囲のコンテナー数を示します。

表 918. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_range	基本

使用法 このエレメントは、DMS 表スペースにのみ適用できます。

range_number 範囲番号

この値は、表スペース・マップ内にある 1 つの範囲の番号を示します。

表 919. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_range	基本

使用法 このエレメントは、DMS 表スペースにのみ適用できます。

range_offset 範囲オフセット

1 つの範囲が属するストライプ・セット開始点のストライプ 0 からのオフセット。

表 920. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_range	基本

使用法 このエレメントは、DMS 表スペースにのみ適用できます。

range_start_stripe 開始ストライプ

この値は、1 つの範囲内の最初のストライプの番号を示します。

表 921. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_range	基本

使用法 このエレメントは、DMS 表スペースにのみ適用できます。

range_stripe_set_number ストライプ・セット番号

この値は、中に 1 つ範囲が含まれているストライプ・セットを示します。

表 922. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_range	基本

使用法 このエレメントは、DMS 表スペースにのみ適用できます。

reclaimable_space_enabled - 再利用可能なスペースが有効な標識 : モニター・エレメント

再利用可能ストレージで表スペースが使用可能な場合、このモニター・エレメントは値 1 を戻します。それ以外の場合は、値 0 を戻します。

表 923. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

rej_curs_blk リジェクトされたブロック・カーソル要求

サーバーでの入出力ブロック要求がリジェクトされて、要求が非ブロック化入出力に変換された回数。

エレメント ID
rej_curs_blk

エレメント・タイプ
カウンター

表 924. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

表 925. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	-

使用法 多数のカーソル・ブロッキング・データがあると、通信ヒープが満杯になることがあります。このヒープが満杯になると、エラーは戻されません。その代わりに、カーソルをブロックするための入出力ブロックが割り振られなくなります。カーソルがデータをブロックできない場合は、パフォーマンスに影響が現れます。

多数のカーソルがデータ・ブロックを実行できない場合は、次のようにしてパフォーマンスを改善できます。

- *query_heap* データベース・マネージャー構成パラメーターのサイズを大きくする。

rem_cons_in - データベース・マネージャーへのリモート接続 :

モニター中のデータベース・マネージャーのインスタンスに対してリモート・クライアントから開始された接続の現在の数。

表 926. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

リモート・クライアントからこのインスタンス内のデータベースへの接続数を示します。この値は頻繁に変化するため、システム使用量の現実的な値を得るためには、長期にわたり特定のインターバルを設けてサンプルを採取する必要があります。この数値には、データベース・マネージャーと同じインスタンスで開始したアプリケーションは含まれません。

これらのエレメントと local_cons モニター・エレメントを組み合わせて使用すると、**max_coordagents** および **max_connections** 構成パラメーターの設定値を調整するときに利用できます。

rem_cons_in_exec - データベース・マネージャーで実行中のリモート接続 :

モニター中のデータベース・マネージャー・インスタンス内のデータベースに現在接続していて、作業単位を現在処理しているリモート・アプリケーションの数。

表 927. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法

この数値は、データベース・マネージャーで実行中の並行処理のレベルを判別するときに利用できます。この値は頻繁に変化するため、システム使用量の現実的な値を得るためには、長期にわたり特定のインターバルを設けてサンプルを採取する必要があります。この数値には、データベース・マネージャーと同じインスタンスで開始したアプリケーションは含まれません。

このエレメントと local_cons_in_exec モニター・エレメントを組み合わせて使用すると、**max_coordagents** 構成パラメーターの設定値を調整するときに利用できます。

max_coordagents を AUTOMATIC に設定する場合は、調整を加える必要はありません。 **max_coordagents** を AUTOMATIC に設定せず、rem_cons_in_exec および local_cons_in_exec の合計が **max_coordagents** に近い場合は、**max_coordagents** の値を増やしてください。

remote_lock_time リモート・ロック時間

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時か、またはデータベース・モニター・カウンターの最後のリセット時のどちらか最近の時点以降に、このフェデレーテッド・サーバー・インスタンス上で実行されているすべてのアプリケーションまたは単一アプリケーションからのリモート・ロックに、このデータ・ソースが要した合計時間が含まれています (ミリ秒単位)。応答時間

は、フェデレーテッド・サーバーがデータ・ソースにリモート・ロックをサブミットしてからフェデレーテッド・サーバーがデータ・ソースでリモート・ロックを解放するまでの時間です。

表 928. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	タイム・スタンプ
アプリケーション	appl_remote	タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

このエレメントを使用すると、このデータ・ソースでリモート・ロックに要した実際の時間を判別できます。

remote_locks リモート・ロック数

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時か、またはデータベース・モニター・カウンターの最後のリセット時のどちらか最近の方の時点以降に、いずれかのアプリケーションに代わってフェデレーテッド・サーバーがこのデータ・ソースで呼び出したリモート・ロックの合計数が含まれています。

表 929. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、データ・ソースでリモート側で行われたリモート・ロックの数を判別できます。

reorg_completion 再編成完了フラグ

マルチディメンション・クラスタリング (MDC) 表からのエクステントの再利用を含む、表の再編成の成否を示す標識です。 パーティション表の場合、データ・パーティションの完了状況も示します。

表 930. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

使用法 表再編成処理またはデータ・パーティション再編成処理が正常に終了すると、このエレメントの値は 0 になります。表再編成処理またはデータ・パーティション再編成処理が失敗すると、このエレメントの値は -1 になります。成功および失敗の値は、sqlmon.h で次のように定義されています。

- 成功: SQLM_REORG_SUCCESS
- 失敗: SQLM_REORG_FAIL

表再編成が異常終了した場合は、履歴ファイルを使用して、警告やエラーなどの診断情報を参照してください。このデータにアクセスするには、LIST HISTORY コマンドを使用します。パーティション表では、完了状況はデータ・パーティションごとに示されます。パーティション表での索引の再作成が失敗した場合、失敗した状況がすべてのデータ・パーティションで更新されます。詳しい診断情報については、管理通知ログを参照してください。

reorg_current_counter 再編成の進行状況

完了した再編成の量を示す進行状況の単位。この値が示す進行の量は、行われる表再編成の合計量を示す reorg_max_counter の値に関連しています。

表 931. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

使用法

次の公式を使用して、完了した表再編成のパーセンテージを判別することができます。

```
表再編成の進行状況 = reorg_current_counter / reorg_max_counter * 100
```

reorg_end 表再編成終了時刻

表再編成の (マルチディメンション・クラスタリング (MDC) 表からエクステントを再利用するための再編成を含む) 終了時刻です。 パーティション表の場合、それぞれのデータ・パーティション再編成の終了時刻も示します。

表 932. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

reorg_index_id 表の再編成に使用される索引

表の再編成に使用されている索引。

表 933. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

reorg_long_tbspc_id - 長いオブジェクトが再編成される表スペース : モニター・エレメント

任意の長いオブジェクト (LONG VARCHAR または LOB データ) が再編成される表スペース。パーティション化されている表の場合、それぞれのパーティションの LONG VARCHAR と LOB が再編成される表スペースです。

表 934. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

reorg_max_counter 再編成の合計量

再編成において行われる作業 (マルチディメンション・クラスタリング (MDC) 表からエクステントを再利用するための再編成を含む) の合計量を示す値です。 この値を、完了した作業の量を示す reorg_current_counter とともに使用して、再編成の進行状況を判別することができます。

表 935. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

reorg_max_phase 再編成の最大フェーズ数

再編成処理のときに発生する再編成フェーズの最大数。これはクラシック (オフライン) 再編成にのみ適用されます。値の範囲は 2 から 4 です ([SORT], BUILD, REPLACE,[INDEX_RECREATE])。この値はマルチディメンション・クラスタリング (MDC) 表からエクステントを再利用するための再編成で行われる作業の合計量を示す場合もあります。 そのような再編成が行われる場合、この値は 3 (SCAN、DRAIN、RELEASE) となります。

表 936. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

reorg_phase - 表の再編成のフェーズ : モニター・エレメント

表の再編成フェーズを示します。パーティション表の場合、それぞれのデータ・パーティションの再編成フェーズも示します。これはオフライン表再編成にのみ適用されます。

表 937. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

使用法

パーティション表の場合、再編成はデータ・パーティション単位でデータ・パーティション上で行われます。クラシック表再編成の場合は、次のフェーズがあります (sqlmon.h ファイルでの対応する定義と共にフェーズをリストしています)。

- ソート: SQLM_REORG_SORT
- ビルド: SQLM_REORG_BUILD
- 置換: SQLM_REORG_REPLACE
- 索引の再作成: SQLM_REORG_INDEX_RECREATE
- ディクショナリーのビルド: SQLM_REORG_DICT_SAMPLE

パーティション表の場合、そのデータ・パーティションの置換フェーズ後に直接、パーティション化索引の索引の再作成フェーズに入ることがあります (パーティション化索引がある場合)。すべてのデータ・パーティションで前のフェーズすべてが正常に完了してからでなければ、**reorg_phase** エレメントは索引の再作成フェーズを示しません。

XDA オブジェクト圧縮時の XML データ再編成フェーズで、表の XML ストレージ・オブジェクトの再編成が行われます。 XML ディクショナリーのビルド・フェーズで、XML ストレージ・オブジェクト用コンプレッション・ディクショナリーの作成が試みられます。 XDA オブジェクト圧縮の場合は、次の 2 つのフェーズがあります。

- XML 再編成: SQLM_REORG_XML_DATA
- XML ディクショナリーのビルド: SQLM_REORG_XML_DICT_SAMPLE

エクステントの再利用が行われているパーティション表の場合は、次のフェーズがあります。

- スキャン: SQLM_REORG_SCAN
- ドレーン: SQLM_REORG_DRAIN
- リリース: SQLM_REORG_RELEASE

reorg_phase_start 表再編成フェーズ開始時刻

表再編成または再利用のための再編成のフェーズの開始時刻。 パーティション表の場合、それぞれのデータ・パーティションの再編成フェーズの開始時刻も示します。索引再作成フェーズの時、すべてのデータ・パーティションでは非パーティション索引のデータ・グループは同時に更新されます。

表 938. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

reorg_rows_compressed - 圧縮行数

再編成中に表で圧縮される行の数。

表 939. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

使用法 再編成中に表で圧縮される行の数の連続したカウント。圧縮されないレコードもあります (レコード・サイズが最小レコード長より小さい場合)。

この行数はデータ圧縮の有効性を測るものではないということに注意してください。これは圧縮基準を満たすレコードの数を示しているにすぎません。

reorg_rows_rejected_for_compression - 圧縮がリジェクトされる行

レコード長が最小レコード長以下であったために再編成中に圧縮されなかった行数。

表 940. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

使用法 レコードが最小レコード長以下の場合は圧縮されません。リジェクトされる行数は、この圧縮要件を満たせないこうしたレコードの連続したカウントを反映しています。

reorg_start 表再編成開始時刻

表再編成の (マルチディメンション・クラスタリング (MDC) 表からエクステントを再利用するための再編成を含む) 開始時刻です。 パーティション表の場合、それぞれのデータ・パーティション再編成の開始時刻も示します。

表 941. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

reorg_status 表再編成の状況

インプレース (オンライン) 表またはデータ・パーティション・レベル再編成の状況。これはクラシック (オフライン) 表再編成には適用できません。

表 942. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

使用法 インプレース表またはデータ・パーティション再編成は、次の状態のいずれかになります (各状態は sqlmon.h での対応する定義とともに示しています)。

- 開始/再開: SQLM_REORG_STARTED
- 休止: SQLM_REORG_PAUSED

- 停止: SQLM_REORG_STOPPED
- 完了: SQLM_REORG_COMPLETED
- 切り捨て: SQLM_REORG_TRUNCATE

エクステントを再利用するインプレース表またはデータ・パーティション再編成は、次の状態のいずれかになります。

- 開始: SQLM_REORG_STARTED
- 停止: SQLM_REORG_STOPPED
- 完了: SQLM_REORG_COMPLETED

reorg_tbspc_id - 表またはデータ・パーティションが再編成される表スペース

表が再編成される表スペース。パーティション化されている表の場合、それぞれのデータ・パーティションが再編成される表スペースも示します。

表 943. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

reorg_type 表再編成の属性

表再編成の属性設定値。

表 944. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

使用法 属性設定値として次のものがあります。各属性設定は、 db2ApiDf.h で定義されるビット・フラグ値に基づいています。

- 書き込みアクセスの許可: DB2REORG_ALLOW_WRITE
- 読み取りアクセスの許可: DB2REORG_ALLOW_READ
- アクセスを許可しない: DB2REORG_ALLOW_NONE
- 索引スキャンを介した再クラスタリング: DB2REORG_INDEXSCAN
- LONG フィールド LOB データの再編成: DB2REORG_LONGLOB
- 表の切り捨てなし: DB2REORG_NOTRUNCATE_ONLINE
- コンプレッション・ディクショナリーの置換: DB2REORG_RESET_DICTIONARY
- コンプレッション・ディクショナリーの維持: DB2REORG_KEEP_DICTIONARY
- エクステントの再利用: DB2REORG_RECLAIM_EXTS

前記の属性設定値に加えて、 GET SNAPSHOT FOR TABLES コマンドの CLP 出力に以下の属性が示されます。これらの属性設定値は、他の属性設定値や表再編成に関するモニター・エレメントの値に基づいています。

- 再クラスタリング: reorg_index_id モニター・エレメントの値がゼロ以外の場合は、表再編成処理はこの属性を持ちます。
- 再利用: reorg_index_id モニター・エレメントの値がゼロの場合は、表再編成処理はこの属性を持ちます。
- インプレース表再編成: reorg_status モニター・エレメントの値が非 NULL の場合は、インプレース (オンライン) 再編成方式が使用されています。
- 表の再編成: reorg_phase モニター・エレメントの値が非 NULL の場合は、クラシック (オフライン) 再編成方式が使用されています。
- 表スキャンを介した再クラスタリング: DB2REORG_INDEXSCAN フラグが設定されていない場合は、表再編成処理はこの属性を持ちます。
- データのみの再編成: DB2REORG_LONGLOB フラグが設定されていない場合は、表再編成処理はこの属性を持ちます。

reorg_xml_regions_compressed – 圧縮された XML 領域 : モニター・エレメント

表の再編成プロセス時に圧縮された XML 領域の数。

表 945. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

reorg_xml_regions_rejected_for_compression – 圧縮を拒否された XML 領域 : モニター・エレメント

表の再編成プロセス時に圧縮されなかった XML 領域の数。

表 946. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table_reorg	基本

request_exec_time_avg 要求の平均実行時間 : モニター・エレメント

最後のリセット以降に、このサービス・サブクラスに関連付けられた要求の実行時間の算術平均。内部的に追跡されている平均がオーバーフローすると、値 -2 が返されます。サービス・サブクラスの COLLECT AGGREGATE REQUEST DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。単位はミリ秒です。

REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、request_exec_time_avg 平均は再マップに関係した部分要求をカウントします。

表 947. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-

使用法

この統計を使用すると、このサービス・サブクラス中のデータベース・パーティション上での要求ごとの平均処理時間を即時に知ることができます。

この平均を使用して、要求の実行時間のヒストグラムに使用されるヒストグラム・テンプレートが適切かどうかを判別することもできます。要求の実行時間のヒストグラムから要求の平均実行時間を計算してください。計算した平均をこのモニター・エレメントと比較してください。計算した平均が、このモニター・エレメントによって報告される真の平均から大きく外れるようなら、データにより適切なビン値のセットを使用する、要求の実行時間のヒストグラムに関するヒストグラム・テンプレートに変更することを考慮してください。

rf_log_num ロールフォワードされているログ

処理中のログ。

エレメント ID
rf_log_num

エレメント・タイプ
情報

表 948. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	rollforward	基本

使用法 ロールフォワードが進行中の場合、このエレメントは関係するログを示します。

rf_status ログ・フェーズ

リカバリーの状況。

エレメント ID
rf_status

エレメント・タイプ
情報

表 949. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	rollforward	基本

使用法 このエレメントはリカバリーの進行状況を示します。リカバリーが取り消し (ロールバック) フェーズにあるのか、あるいは再実行 (ロールフォワード) フェーズにあるのかを示します。

rf_timestamp ロールフォワード・タイム・スタンプ

最後にコミットしたトランザクションのタイム・スタンプ。

エレメント ID
rf_timestamp

エレメント・タイプ
timestamp

表 950. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	rollforward	タイム・スタンプ

使用法 ロールフォワードが進行中の場合、これはロールフォワード・リカバリーが処理した、最後にコミットされたトランザクションのタイム・スタンプです。これは、どの程度ロールフォワード操作が進行したかを示す指標になります。

rf_type ロールフォワード・タイプ

進行中のロールフォワードのタイプ。

エレメント ID
rf_type

エレメント・タイプ
情報

表 951. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	rollforward	基本

使用法 データベース・レベルと表スペース・レベルのどちらでリカバリーが起きているかを示す標識です。

rollback_sql_stmts 試行されたロールバック・ステートメント

試行された SQL ROLLBACK ステートメントの合計数。

エレメント ID
rollback_sql_stmts

エレメント・タイプ
カウンター

表 952. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
データベース	dbase_remote	基本
アプリケーション	appl	基本
アプリケーション	appl_remote	基本

表 952. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	基本
DCS アプリケーション	dcs_appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 953. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 ロールバックは、アプリケーション要求、デッドロック、またはエラー状態の結果として起こります。このエレメントでは、アプリケーションが発行したロールバック・ステートメント**のみ**カウントされます。

アプリケーション・レベルでは、このエレメントはアプリケーションのデータベース・アクティビティー・レベルとその他のアプリケーションとの競合の量を判別するのに役立ちます。データベース・レベルでは、データベース内のアクティビティーの量とデータベース上でのアプリケーション間の競合の量を判別できます。

注: ロールバック・アクティビティーが多くなるとデータベースのスループットが低下するので、ロールバックの回数を最小限にとどめてください。

次の項目を合計すると、作業単位の合計数も計算できます。

```
  commit_sql_stmts
+ int_commits
+ rollback_sql_stmts
+ int_rollbacks
```

rolled_back_agent_id ロールバックされたエージェント

デッドロックが発生したときにロールバックされたエージェント。

エレメント ID

rolled_back_agent_id

エレメント・タイプ

情報

表 954. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	event_deadlock	-

使用法 システム管理者はこの情報を基にして、更新を完了できなかったアプリケーションや再始動する必要があるアプリケーションを判別できます。

rolled_back_appl_id ロールバック・アプリケーション

デッドロックが発生したときにロールバックされたアプリケーション ID。

エレメント ID

rolled_back_appl_id

エレメント・タイプ

情報

表 955. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	event_deadlock	-

使用法 システム管理者はこの情報を基にして、更新を完了できなかったアプリケーションや再始動する必要があるアプリケーションを判別できます。

rolled_back_participant_no ロールバック参加アプリケーション : モニター・エレメント

ロールバックされたアプリケーションを識別する参加者の番号。

表 956. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
デッドロック[1]	event_deadlock	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

システム管理者はこの情報を基にして、更新を完了できなかったアプリケーションや始動する必要があるアプリケーションを判別できます。

rolled_back_sequence_no ロールバックされたシーケンス番号

デッドロックが発生したときにロールバックされたアプリケーションのシーケンス番号。

エレメント ID

rolled_back_sequence_no

エレメント・タイプ

情報

表 957. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	event_deadlock	-

使用法 システム管理者はこの情報を基にして、更新を完了できなかったアプリケーションや再始動する必要があるアプリケーションを判別できます。

root_node_splits - ルート・ノードの分割 : モニター・エレメント

挿入操作時に索引のルート・ノードが分割された回数。

表 958. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

routine_id - ルーチン ID : モニター・エレメント

このモニター・エレメントは、固有のルーチン ID です。アクティビティーがルーチンの一部ではない場合、ゼロが戻されます。

表 959. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

rows_deleted - 削除行数 : モニター・エレメント

これは、試行された行の削除の数です。

表 960. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLE 表関数 - 表メトリックの取得	常に収集される

表 961. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
データベース	dbase_remote	基本
アプリケーション	appl	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 962. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 このエレメントを使用すると、データベース内の現在のアクティビティーのレベルがわかります。

このカウントには、**int_rows_deleted** モニター・エレメントでカウントされる試行回数は含まれません。

rows_fetched フェッチ行数 : モニター・エレメント

表から読み取られた行の数。

このモニター・エレメントは、**rows_read** モニター・エレメントの別名です。

注: このモニター・エレメントは、この情報を記録する対象としたデータベース・パーティションにおける値のみを報告します。 DPF システムでは、これらの値はアクティビティー全体の合計を正しく反映しない場合があります。

エレメント ID
rows_fetched

エレメント・タイプ
カウンター

表 963. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	ステートメント

使用法

詳しくは、**rows_read** モニター・エレメントを参照してください。

rows_inserted - 挿入行数 : モニター・エレメント

試行された行の挿入の数。

表 964. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLE 表関数 - 表メトリックの取得	常に収集される

表 965. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 965. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	基本
アプリケーション	appl	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 966. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 このエレメントを使用すると、データベース内の現在のアクティビティーのレベルがわかります。

フェデレーテッド・システムの場合は、状況によってはフェデレーテッド・サーバーが INSERT FROM SUBSELECT をデータ・ソースにプッシュできるので、 INSERT ステートメントごとに複数の行を挿入できます。

このカウントには、**int_rows_inserted** モニター・エレメントでカウントされる試行回数は含まれません。

rows_modified 変更行数 : モニター・エレメント

挿入、更新、または削除された行数。

このモニター・エレメントは、**rows_written** モニター・エレメントの別名です。

表 967. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 967. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 968. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
アクティビティー	event_activity	ステートメント

使用法

詳しくは、**rows_written** モニター・エレメントを参照してください。

rows_read - 読み取り行数 : モニター・エレメント

表から読み取られた行の数。

表 969. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 969. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	常に収集される
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 970. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
表	table	表
アプリケーション	appl	基本
アプリケーション	stmt	基本
アプリケーション	subsection	ステートメント
動的 SQL	dynsql	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 971. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
接続	event_conn	-
表	event_table	-
ステートメント	event_stmt	-

表 971. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
トランザクション	event_xact	-

使用法

このエレメントを使用すると、使用率が高く、新たな索引を追加する必要があるような表を識別できます。不要な索引の保守を回避するには、SQL EXPLAIN ステートメントを使用して、パッケージが索引を使用するかどうかを判別します。

このカウントは、呼び出し側のアプリケーションに戻された行数ではありません。結果セットを戻すために、読み取りが必要になった行数です。例えば、次のステートメントを使用すると、アプリケーションに戻されるのは 1 行ですが、平均給与を判別するために多数の行が読み取られます。

```
SELECT AVG(SALARY) FROM USERID.EMPLOYEE
```

このカウントには、**overflow_accesses** モニター・エレメントの値が含まれます。ただし、索引アクセスは含まれません。つまり、アクセス・プランが索引アクセスだけを使用して、実際の行を見るために表の読み取りを行わなければ、**rows_read** モニター・エレメントの値は増えません。

rows_returned 戻り行数 : モニター・エレメント

選択されてアプリケーションに戻された行数。このエレメントは、アクティビティー・レコードが部分的な場合 (例えば、アクティビティーがまだ実行中に収集された場合、またはメモリーの制約のために完全なアクティビティー・レコードをイベント・モニターに書き込むことができなかったとき) に、値が 0 になります。

このモニター・エレメントは、**fetch_count** モニター・エレメントの別名です。

表 972. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 972. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 973. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
アクティビティー	event_activity	-

使用法

このエレメントを使用すると、アプリケーションに戻される行数のしきい値を判別する助けになります。または、そのようなしきい値が正しく構成され、作動しているかを検証するために使用できます。

rows_returned_top 実際の戻り行数の最上位 : モニター・エレメント

サービス・クラスまたは作業クラスでの、すべてのネスト・レベルにおける DML アクティビティーの実際の戻り行数の最高水準点。サービス・クラスでは、サービス・クラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。

サービス・クラスの場合、REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーが完了したサービス・サブクラスの rows_returned_top 最高水準点のみが更新されます。アクティビティーがマップされたサービス・サブクラスでも、アクティビティーがそこで完了しなかった場合、そのサービス・サブクラスの最高水準点は何も影響を受けません。

表 974. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

このエレメントを使用して、収集された時間間隔にサービス・クラス、ワークロード、または作業クラス用のパーティションで到達した DML アクティビティーの実際の戻り行数の最大数を調べることができます。

rows_selected 選択行数

これは、選択されてアプリケーションに戻された行の数です。

表 975. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
データベース	dbase_remote	基本
アプリケーション	appl	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 976. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 このエレメントを使用すると、データベース内の現在のアクティビティーのレベルがわかります。

このエレメントには、 COUNT(*) や結合などのアクションで読み込まれた行のカウントは含まれません。

フェデレーテッド・システムの場合は、データ・ソースからフェデレーテッド・サーバーに行を戻すのに要する平均時間を計算できます。

```
平均時間 = 戻された行数 / 照会応答合計時間
```

この結果を使用すると、SYSCAT.SERVERS 内の CPU 速度または通信速度などのパラメーターを変更できます。これらのパラメーターを変更すると、オプティマイザーが要求をデータ・ソースに送信するかしないかに影響を与えます。

注: モニター対象のゲートウェイが DB2 データベース・バージョン 7.2 以前のものである場合は、このエレメントはスナップショット・モニターの dcs_dbase および dcs_appl 論理データ・グループで収集されます。

rows_updated - 更新行数 : モニター・エレメント

これは、試行された行の更新の数です。

表 977. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLE 表関数 - 表メトリックの取得	常に収集される

表 978. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
データベース	dbase_remote	基本
アプリケーション	appl	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 979. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 このエレメントを使用すると、データベース内の現在のアクティビティーのレベルがわかります。

この値には、**int_rows_updated** モニター・エレメントでカウントされる更新は含まれません。ただし、複数の UPDATE ステートメントにより更新された行は、更新ごとにカウントされます。

rows_written 書き込み行数

これは、表内で変更 (挿入、削除、または更新) された行の数です。

表 980. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本
アプリケーション	appl	基本

表 980. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	基本
アプリケーション	サブセクション	ステートメント
動的 SQL	dynsql	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 981. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	-
表	event_table	-
ステートメント	event_stmt	-
トランザクション	event_xact	-

使用法 表レベルの情報で数値が高い場合は表の使用率が高いことを示しているため、 Run Statistics (RUNSTATS) ユーティリティーを使用して、この表に使用されるパッケージの効率を維持する必要があります。

アプリケーション接続およびステートメントでは、一時表で挿入、更新および削除があった行数がこのエレメントに含まれます。

アプリケーション、トランザクション、およびステートメントのレベルでこのエレメントを使用すると、相対的アクティビティー・レベルを解析したり、調整できる項目を見つけるのに便利です。

rqsts_completed_total - 完了した要求の合計 : モニター・エレメント

実行された要求の合計数。アプリケーション要求と内部要求の両方を含みます。サービス・サブクラスでは、このモニター・エレメントは要求が完了した場合にのみ更新されます。異なるサービス・サブクラス間を要求が移動した場合、2 回カウントされることはありません。

表 982. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE

表 982. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 983. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

sc_work_action_set_id サービス・クラス作業アクション・セット ID : モニター・エレメント

このアクティビティーがサービス・クラス有効範囲の作業クラスにカテゴリー化されている場合、このモニター・エレメントは、この作業クラスが所属する作業クラス・セットに関連した作業アクション・セットの ID を表示します。それ以外の場合、このモニター・エレメントは 0 の値を表示します。

表 984. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
WLM_GET_ACTIVITY_DETAILS_COMPLETE 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

表 985. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

使用法

このエレメントと **sc_work_class_id** エレメントを組み合わせて使用すると、アクティビティーのサービス・クラス作業クラスが存在する場合にはそれを一意的に識別できます。

sc_work_class_id サービス・クラス作業クラス ID : モニター・エレメント

このアクティビティーがサービス・クラス有効範囲の作業クラスにカテゴリー化されている場合、このモニター・エレメントは、このアクティビティーに割り当てられた作業クラスの ID を表示します。それ以外の場合、このモニター・エレメントは 0 の値を表示します。

表 986. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
WLM_GET_ACTIVITY_DETAILS_COMPLETE 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

表 987. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

使用法

このエレメントと **sc_work_action_set_id** エレメントを組み合わせて使用すると、アクティビティーのサービス・クラス作業クラスが存在する場合にはそれを一意的に識別できます。

sec_log_used_top 使用された最大 2 次ログ・スペース

使用された 2 次ログ・スペースの最大量 (バイト単位)。

表 988. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 989. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 このエレメントと *sec_logs_allocated* および *tot_log_used_top* を組み合わせて使用すると、 2 次ログへの現在の依存度が示されます。この値が高い場合は、より大きなログ・ファイル、またはより多くの 1 次ログ・ファイル、あるいはアプリケーション内でより頻度の高い COMMIT ステートメントが必要になります。

結果として、次の構成パラメーターの調整が必要になります。

- logfilsiz
- logprimary
- logsecond
- logretain

データベースに 2 次ログ・ファイルがまったくない場合は、この値はゼロになります。定義されていない場合もゼロになります。

注: データベース・システム・モニター情報はバイト単位で示されますが、構成パラメーターは各 4K バイトのページ単位で設定されます。

sec_logs_allocated 現在割り振られている 2 次ログ

データベースで現在使用されている 2 次ログ・ファイルの合計数。

表 990. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 このエレメントと *sec_log_used_top* および *tot_log_used_top* を組み合わせて使用すると、 2 次ログへの現在の依存度が分かります。この値が常に高い場合は、より大きなログ・ファイル、またはより多くの 1 次ログ・ファイル、あるいはアプリケーション内でより頻度の高い COMMIT ステートメントが必要になります。

結果として、次の構成パラメーターの調整が必要になります。

- logfilsiz
- logprimary
- logsecond
- logretain

section_env セクション環境 : モニター・エレメント

アクティビティーのセクションの詳細を示すハンドル。

表 991. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activitystmt	-

使用法

このエレメントは、このレコードで記述されているアクティビティーのセクション情報を抽出するための、将来の IBM ツールで使用されます。

section_number - セクション番号 : モニター・エレメント

現在処理中または最後に処理された静的 SQL ステートメントのパッケージにある内部セクション番号。

表 992. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	常に収集される

表 993. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント
DCS ステートメント	dcs_stmt	ステートメント

表 994. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック[1]	event_detailed_dlconn	-
ステートメント	event_stmt	-
アクティビティー	event_activitystmt	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

静的 SQL ステートメントの場合は、このエレメントと **creator**、**package_version_id**、および **package_name** モニター・エレメントを組み合わせて使用すると、次のサンプル照会を使用して、SYSCAT.STATEMENTS システム・カタログ表を照会し、静的 SQL ステートメント・テキストを取得できます。

```
SELECT SEQNO, SUBSTR(TEXT,1,120)
       FROM SYSCAT.STATEMENTS
       WHERE PKGNAME   = 'package_name' AND
             PKGSCHEMA = 'creator'      AND
             VERSION = 'package_version_id' AND
             SECTNO    = section_number
       ORDER BY SEQNO
```

注: 静的ステートメント・テキストを取得するときは、システム・カタログ表にこの照会をするとロックの競合を起こすことがあるので注意してください。この照会を使用するのは、できるだけデータベースに対するその他のアクティビティーが少ないときだけにしてください。

section_type - セクション・タイプ標識 : モニター・エレメント

SQL ステートメント・セクションが動的であるかまたは静的であるかを示します。

表 995. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

使用法

このモニター・エレメントの可能な値は以下のとおりです。

- D: 動的
- S: 静的

select_sql_stmts 実行された選択 SQL ステートメント

実行された SQL SELECT ステートメントの数。

表 996. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
データベース	dbase_remote	基本
表スペース	tablespace	基本
アプリケーション	appl	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 997. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 このエレメントを使用すると、アプリケーション・レベルまたはデータベース・レベルのデータベース・アクティビティーのレベルを判別できます。

次の公式を使用すると、すべてのステートメントに対する SELECT ステートメントの比率を計算できます。

```
  select_sql_stmts
/ ( static_sql_stmts
  + dynamic_sql_stmts )
```

この情報は、アプリケーションのアクティビティーおよびスループットの分析に役立ちます。

select_time 照会応答時間

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時か、またはデータベース・モニター・カウンターの最後のリセット時のどちらか最近の方の時点以降に、このフェデレーテッド・サーバー・インスタンス上で実行されているすべてのアプリケーションまたは単一アプリケーションからの照会に対して、このデータ・ソースが応答に要した合計時間が含まれています (ミリ秒単位)。

注: 照会ブロッキングが行われるため、フェデレーテッド・サーバーが行の読み取りを試行しても、その通信がすべて処理されるとは限りません。取得を要求した次の行は、戻される行のブロックに入っている可能性があります。そのため、照会応答合計時間は、データ・ソースでの処理を示すとは限らず、実際にはデータ・ソースまたはクライアントでの処理を示します。

表 998. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	タイム・スタンプ
アプリケーション	appl_remote	タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

このエレメントを使用すると、このデータ・ソースのデータを待機した実際の時間を判別できます。この情報は、キャパシティー・プランニング、CPU のチューニング、および SYSCAT.SERVERS の通信速度を調整するときに役に立ちます。これらのパラメーターを変更すると、オプティマイザーが要求をデータ・ソースに送信するかしないかに影響を与えます。

応答時間は、フェデレーテッド・サーバーがデータ・ソースから行を要求してからフェデレーテッド・サーバーがその行を利用できるようになるまでの時間です。

sequence_no シーケンス番号 : モニター・エレメント

作業単位が終了するごとに (つまり、 COMMIT または ROLLBACK が作業単位を終了するごとに) この ID が増加します。 **appl_id** と **sequence_no** を使用してトランザクションを一意的に識別します。

表 999. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	基本
DCS アプリケーション	dcs_appl_info	基本

表 1000. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	-
接続	event_connheader	-
ステートメント	event_stmt	-

表 1000. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
トランザクション	event_xact	-
デッドロック	event_dlconn	-
詳細付きデッドロック	event_detailed_dlconn	-
詳細付きデッドロック履歴	event_detailed_dlconn	-
詳細付きデッドロック履歴	event_stmt_history	-
詳細付きデッドロック履歴値	event_detailed_dlconn	-
詳細付きデッドロック履歴値	event_stmt_history	-

sequence_no_holding_lk ロックを保持しているシーケンス番号

このアプリケーションが取得のために待機しているオブジェクトのロックを保留しているアプリケーションのシーケンス番号。

エレメント ID

sequence_no_holding_lk

エレメント・タイプ

情報

表 1001. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本
ロック	appl_lock_list	基本

表 1002. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	event_dlconn	-
詳細付きデッドロック	event_detailed_dlconn	-

使用法 この ID と appl_id を組み合わせて使用すると、このアプリケーションが取得しようと待機しているオブジェクトのロックを保留しているトランザクションを一意的に識別できます。

server_db2_type モニター対象 (サーバー) ノードのデータベース・マネージャーのタイプ

モニター中のデータベース・マネージャーのタイプを識別します。

表 1003. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	collected	基本

使用法 データベース・マネージャーについて、次の構成タイプの 1 つが含まれます。

API シンボリック定数
コマンド行プロセッサー出力

sqlf_nt_server
ローカル・クライアントおよびリモート・クライアントを指定したデータベース・サーバー

sqlf_nt_stand_req
ローカル・クライアントを指定したデータベース・サーバー

API シンボリック定数は、組み込みファイルの *sqlutil.h* に定義されています。

server_instance_name サーバー・インスタンス名

スナップショットが取られるデータベース・マネージャー・インスタンスの名前。

エレメント ID
server_instance_name

エレメント・タイプ
情報

表 1004. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	collected	基本

表 1005. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
イベント・ログ・ヘッダー	event_log_header	-

使用法 複数のデータベース・マネージャーのインスタンスが同一のシステム上にある場合、このデータ項目は、スナップショット呼び出しが行われたインスタンスを一意的に識別するために使用されます。この情報は、モニター出力を後で解析するためにファイルやデータベースに保管しており、データベース・マネージャー の各インスタンスごとにデータを区別する必要がある場合に役立ちます。

server_platform サーバーのオペレーティング・システム

データベース・サーバーが稼働中のオペレーティング・システム。

エレメント ID
server_platform

エレメント・タイプ
情報

表 1006. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 1007. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 このエレメントを使用して、リモート・アプリケーションの問題判別を行えます。このフィールドの値は、ヘッダー・ファイルの *sqlmon.h* にあります。

server_prdid - サーバー製品/バージョン ID

サーバー上で実行中の製品とバージョン。

表 1008. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	collected	基本

表 1009. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
イベント・ログ・ヘッダー	event_log_header	-

使用法 PPPVVRRM の形式になっています。各部分の定義は次のとおりです。

PPP SQL です。

VV 2 桁でバージョン番号を示します (バージョン番号が 1 桁の場合には、高位の桁は 0 になります)。

RR 2 桁でリリース番号を示します (リリース番号が 1 桁の場合には、高位の桁は 0 になります)。

M 1 文字で修正レベルを示します (0-9 または A-Z)。

server_version サーバー・バージョン

情報を戻しているサーバーのバージョン。

表 1010. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	collected	基本

使用法

このフィールドは、データベース・システム・モニター情報を収集しているデータベース・サーバーのレベルを識別します。これにより、アプリケーションは、データを戻しているサーバーのレベルに基づいてデータを解釈することができます。有効な値は以下のとおりです。

SQLM_DBMON_VERSION1
データは、 DB2 バージョン 1 によって戻されました。

SQLM_DBMON_VERSION2
データは、 DB2 バージョン 2 によって戻されました。

SQLM_DBMON_VERSION5
データは、 DB2® Universal Database™ バージョン 5 によって戻されました。

SQLM_DBMON_VERSION5_2
データは、 DB2 Universal Database バージョン 5.2 によって戻されました。

SQLM_DBMON_VERSION6
データは、 DB2 Universal Database バージョン 6 によって戻されました。

SQLM_DBMON_VERSION7
データは、 DB2 Universal Database バージョン 7 によって戻されました。

SQLM_DBMON_VERSION8
データは、 DB2 Universal Database バージョン 8 によって戻されました。

SQLM_DBMON_VERSION9
データは、 DB2 Database for Linux, UNIX, and Windows バージョン 9 によって戻されました。

SQLM_DBMON_VERSION9_5
データは、 DB2 Database for Linux, UNIX, and Windows バージョン 9.5 によって戻されました。

service_class_id サービス・クラス ID : モニター・エレメント

サービス・サブクラスのユニーク ID。ワークロードの場合、この ID はワークロードがマップされた先のサービス・サブクラスを表します。作業単位の場合、この ID は、その作業単位を発行している接続が関連付けられているワークロードのサービス・サブクラス ID を表します。

表 1011. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	常に収集される
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得	常に収集される
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	常に収集される
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	常に収集される
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	常に収集される
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

表 1012. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
ロッキング	-	-
作業単位	-	-
統計	event_histogrambin	-
統計	event_scstats	-

使用法

このエレメントの値は、ビュー SYSCAT.SERVICECLASSES の列 SERVICECLASSID の値と一致します。このエレメントを使用して、サービス・サブクラス名、または別のソースのサービス・サブクラスに関するリンク情報を検索します。例えば、サービス・クラス統計をヒストグラム・ビン・レコードと結合させます。

service_level サービス・レベル

DB2 インスタンスの現在の修正サービス・レベルを示します。

表 1013. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

service_subclass_name サービス・サブクラス名 : モニター・エレメント

サービス・サブクラスの名前。

表 1014. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	常に収集される
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	常に収集される
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

表 1015. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
ロッキング	-	-
作業単位	-	-
アクティビティー	event_activity	-
統計	event_scstats	-
統計	event_qstats	-

使用法

このエレメントを他のアクティビティー・エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。あるいは、他の統計エレメントと一緒に使用すると、サービス・クラスまたはしきい値キューの動作の分析をすることができます。

service_superclass_name サービス・スーパークラス名 : モニター・エレメント

サービス・スーパークラスの名前。

表 1016. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	常に収集される
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	常に収集される
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

表 1017. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	-	-
アクティビティー	event_activity	-
統計	event_scstats	-
統計	event_qstats	-

使用法

このエレメントを他のアクティビティー・エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。あるいは、他の統計エレメントと一緒に使用すると、サービス・クラスまたはしきい値キューの動作の分析をすることができます。

session_auth_id セッション許可 ID : モニター・エレメント

このアプリケーションによって使用されている現在のセッション許可 ID。ワークロード管理アクティビティーのモニターでは、このモニター・エレメントは、アクティビティーがシステムに挿入された時のセッション許可 ID を記述します。

表 1018. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
ロック	appl_lock_list	基本

表 1019. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	-
アクティビティー	event_activity	-
しきい値違反	event_activity	-

使用法

このエレメントは、SQL ステートメントの準備、SQL ステートメントの実行、またはその両方を行うのにどの許可 ID が使用されているかを判別するのに役立ちます。このモニター・エレメントは、ストアード・プロシージャーの実行中に設定されたセッション許可 ID 値は報告しません。

shr_workspace_num_overflows 共有ワークスペースのオーバーフロー回数

割り振られたメモリーの境界から共有ワークスペースがオーバーフローした回数。

注: このモニター・エレメントは、使用しないでください。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 1020. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1021. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 このエレメントと shr_workspace_size_top を組み合わせて使用すると、オーバーフローを防止するのに共有ワークスペースのサイズを大きくする必要があるかどうかを判別できます。共有ワークスペースがオーバーフローすると、パフォーマンスが低下するだけではなく、アプリケーションの共有メモリーから割り振られたほかのヒープでメモリー不足エラーが発生することがあります。

データベース・レベルでは、「共有ワークスペースの最大サイズ」のある共有ワークスペースとして報告された共有ワークスペースがこのエレメントの報告の対象となります。アプリケーション・レベルでは、現行アプリケーションが使用するワークスペースのオーバーフロー回数を示します。

shr_workspace_section_inserts 共有ワークスペース・セクション挿入数

共有ワークスペースへの、アプリケーションによる SQL セクション挿入数。

注: このモニター・エレメントは、使用しないでください。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 1022. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1023. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 実行可能セクションの作業用コピーは、共有ワークスペース内に保管されます。このカウンターは、コピーが使用できなかったために挿入が必要だった場合を示します。

データベース・レベルでは、データベース内のすべての共有ワークスペースを対象に、すべてのアプリケーションでの累計挿入数を示します。アプリケーション・レベルでは、このアプリケーションの共有ワークスペース内にあるすべてのセクションを対象とした累計挿入数を示します。

shr_workspace_section_lookups 共有ワークスペース・セクション検索

共有ワークスペースでの、アプリケーションによる SQL セクション検索数。

注: このモニター・エレメントは、使用しないでください。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 1024. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1025. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 各アプリケーションは、実行可能セクションの作業用コピーがある共有ワークスペースにアクセスできます。

このカウンターは、アプリケーションの特定のセクションを見つけるために共有ワークスペースがアクセスされた回数を示します。データベース・レベルでは、データベース内のすべての共有ワークスペースを対象に、すべてのアプリケーションでの累計検索数を示します。アプリケーション・レベルでは、このアプリケーションの共有ワークスペース内にあるすべてのセクションを対象とした累計検索数を示します。

このエレメントと「共有ワークスペース・セクション挿入数」を組み合わせて使用すると、共有ワークスペースのサイズを調整できます。共有ワークスペースのサイズをコントロールしているのは、 app_ctl_heap_sz 構成パラメーターです。

shr_workspace_size_top 最大共有ワークスペース・サイズ

共有ワークスペースが到達した最大サイズ。

注: このモニター・エレメントは、使用しないでください。このモニター・エレメントを使用しても、エラーは生成されません。そして、有効な値も戻されません。このモニター・エレメントは推奨されておらず、将来のリリースではサポートされなくなる予定です。

表 1026. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

表 1027. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 このエレメントは、データベースが活動化されて以降、データベースでワークロードを実行したときに必要となった共有ワークスペースの最大バイト数を示します。データベース・レベルでは、すべての共有ワークスペースが到達した最大サイズを示します。アプリケーション・レベルでは、現行アプリケーションが使用する共有ワークスペースの最大サイズです。

共有ワークスペースがオーバーフローした場合、このエレメントは、オーバーフロー時に共有ワークスペースが到達した最大サイズになります。このような状態が発生したかどうかを確認するには、「共有ワークスペースのオーバーフロー回数」をチェックしてください。

共有ワークスペースがオーバーフローすると、アプリケーションの共有メモリーにあるほかのエンティティーからメモリーを一時的に借用します。この結果、これらのエンティティーでメモリー不足エラーが発生したり、パフォーマンスが低下することがあります。 APP_CTL_HEAP_SZ を大きくすると、オーバーフローの確率を低くすることができます。

smallest_log_avail_node 使用可能なログ・スペースが最小のノード

このエレメントはグローバル・スナップショットの場合にだけ戻され、使用可能なログ・スペースが最も少ない (バイト数) ノードを示します。

エレメント ID

smallest_log_avail_node

エレメント・タイプ

情報

表 1028. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 このエレメントと appl_id_oldest_xact を組み合わせて使用すると、データベースに十分なログ・スペースがあることを確認できます。グローバル・スナップショットでは、appl_id_oldest_xact、total_log_used、および total_log_available がこのノードの値に対応します。

sort_heap_allocated 割り振られたソート・ヒープの合計

スナップショットが取られたときに、選択したレベルのすべてのソートに割り振られたソート・ヒープ・スペース用のページ数の合計。

表 1029. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

表 1029. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 各ソートに割り振られたメモリー量は、利用可能なソート・ヒープ・サイズの一部だけの場合とすべての場合があります。ソート・ヒープ・サイズは各ソートで利用できるメモリー量を示し、 *sortheap* データベース構成パラメーターに定義されている値です。

1 つのアプリケーションが同時に複数のソートをアクティブにすることができます。例えば、副照会付きの SELECT ステートメントを使用すると、同時に複数のソートが行われる場合があります。

情報は 2 つのレベルで収集できます。

- データベース・マネージャーのレベルでは、データベース・マネージャー内のアクティブなすべてのデータベースのすべてのソートを対象に、割り振られたソート・ヒープ・スペースの合計を示す。
- データベース・レベルでは、1 つのデータベース内のすべてのソートを対象に、割り振られたソート・ヒープ・スペースの合計を示す。

通常のメモリーの見積もりにはソート・ヒープ・スペースは含まれません。過剰なソートが発生している場合は、ソート・ヒープに使用される追加のメモリー量をデータベース・マネージャーを実行するのに必要な基本メモリー量に加える必要があります。一般に、ソート・ヒープが大きくなるほど、ソート効率は高くなります。索引を正しく使用すると、ソートに必要な量を少なくできます。

データベース・マネージャー・レベルに戻された情報は、 *sheapthres* 構成パラメーターの調整に利用できます。エレメントの値が *sheapthres* 以上になっている場合は、 *sortheap* パラメーターに定義されているソート・ヒープをソートで完全に得られていないことを示します。

sort_heap_top ソート専用ヒープの最高水準点

データベース・マネージャーでの専用ソート・メモリーの最高水準点 (4 KB ページ単位)。

表 1030. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	db2	基本

使用法 このエレメントを使用して、 SHEAPTHRES 構成パラメーターが最適な値に設定されているかどうかを判別できます。例えば、この水準点が SHEAPTHRES に近づいたり超えている場合は、おそらく SHEAPTHRES を大きくする必要があります。これは、SHEAPTHRES を超えると専用ソートに与えられるメモリーが常に少なくなり、その結果として逆にシステム・パフォーマンスに影響を与える場合があるためです。

sort_overflows - ソート・オーバーフロー : モニター・エレメント

ソート・ヒープを使い果たし、一時記憶用のディスク・スペースが必要になった可能性のあるソートの合計数。

表 1031. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 1032. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本
アプリケーション	stmt	基本
動的 SQL	dynsql	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1033. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
接続	event_conn	-
ステートメント	event_stmt	-
アクティビティー	event_activity	ステートメント、ソート

使用法

データベース・レベルまたはアプリケーション・レベルでは、この値と **total_sorts** を組み合わせて使用すると、ディスクにオーバーフローしたソートのパーセンテージを計算できます。このパーセンテージが高い場合は、**sortheap** の値を大きくして、データベース構成を調整する必要があります。

ステートメント・レベルでこのエレメントを使用すると、大量のソートを必要とするステートメントを識別できます。このようなステートメントは、さらに調整を行って必要となるソート量を少なくすると効率が上がります。

ソートがオーバーフローすると、ソートにマージ・フェーズが必要となり、データをディスクに書き込む必要がある場合は入出力がさらに必要となるので、オーバーヘッドが増えます。

このエレメントは、1 ステートメント、1 アプリケーション、または 1 つのデータベースにアクセスするすべてのアプリケーションについて情報を提供します。

sort_shrheap_allocated 現在割り振られているソート共有ヒープ

データベースに割り振られている共有ソート・メモリーの合計量。

表 1034. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 このエレメントを使用して、共有ソート・メモリーのしきい値を評価できます。この値が共有ソート・メモリーの現行しきい値より大幅に高いことや低いことが頻繁にある場合は、おそらく、しきい値を調整する必要があります。

注: 「共有ソート・メモリーしきい値」は、 SHEAPTHRES_SHR データベース構成パラメーターが 0 の場合は SHEAPTHRES データベース・マネー

ジャー構成パラメーターの値で決まります。 0 でない場合は SHEAPTHRES_SHR の値で決まります。

sort_shrheap_top ソート共有ヒープの最高水準点

データベース全体の共有ソート・メモリーの最高水準点 (4 KB ページ単位)。

表 1035. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 このエレメントを使用して、 SHEAPTHRES (または SHEAPTHRES_SHR) が最適な値に設定されているかどうかを評価できます。例えば、この最高水準点が常に共有ソート・メモリーしきい値よりも大幅に低い場合は、おそらくこのしきい値を小さくしてデータベースの他の機能にメモリーを解放する必要があります。逆にこの最高水準点が共有ソート・メモリーしきい値に近づき始めたら、そのしきい値を大きくする必要がある場合があります。共有ソート・メモリーしきい値はハード・リミットなので余裕を持たせておくことは重要です。ソート・メモリーの合計量がそのしきい値に達したら、共有ソートは開始できなくなります。

このエレメントは、専用ソート・メモリーの最高水準点と組み合わせて使用すると、共有および専用ソートのしきい値をそれぞれ単独に設定する必要があるかどうかを判別することにも利用できます。SHEAPTHRES_SHR データベース構成オプションの値が 0 の場合は通常、共有ソート・メモリーしきい値は SHEAPTHRES データベース・マネージャー構成オプションの値で決まります。ただし専用ソート・メモリーと共有ソート・メモリーの最高水準点に大きな違いがある場合は、SHEAPTHRES をオーバーライドして、SHEAPTHRES_SHR を共有ソート・メモリーの最高水準点を基にした、より適切な値に設定する必要がある場合があります。

source_service_class_id ソース・サービス・クラス ID : モニター・エレメント

このエレメントのしきい値違反レコードが生成された時に、アクティビティーから再マップしたサービス・サブクラスの ID。 しきい値アクションが REMAP ACTIVITY アクション以外の場合、このエレメントの値はゼロです。

エレメント ID

source_service_class_id

エレメント・タイプ

情報

表 1036. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-

使用法

このエレメントは、アクティビティーが再マップされたサービス・クラスをたどるのに使用できます。 これを使用して、特定のサービス・サブクラスからマップされたアクティビティー数の総計を計算することもできます。

sp_rows_selected ストアード・プロシージャーによって戻された行数

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時点か、またはデータベース・モニター・カウンターの最後のリセット時点以降に、このアプリケーションのストアード・プロシージャーの処理の結果として、データ・ソースからフェデレーテッド・サーバーに送信された行の数が含まれています。

表 1037. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントは複数の目的に使用できます。次の公式を使用すると、ストアード・プロシージャー単位でデータ・ソースからフェデレーテッド・サーバーに送信された平均行数を計算できます。

```
   ストアード・プロシージャー単位の行数
= 戻された行数
/ 呼び出されたストアード・プロシージャーの数
```

このアプリケーションについて、データ・ソースからフェデレーテッド・サーバーに行を戻すときの平均時間も計算できます。

```
平均時間 = ストアード・プロシージャー応答合計時間 / 戻された行数
```

sql_chains 試行された SQL チェーンの数

ステートメント処理中に DB2 Connect ゲートウェイとホストの間で *n* 回のデータ伝送が行われる際の、 SQL ステートメントの数を表します。範囲 *n* は、*num_transmissions_group* エレメントで指定されます。

表 1038. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データ伝送	stmt_transmissions	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

例えば、チェーニングが ON の場合に、 PREP ステートメントと OPEN ステートメントがチェーニングされ、チェーンが合計 2 つの伝送を要する場合は、*sql_chains* は「1」と報告され、 *sql_stmts* は「2」と報告されます。

チェーニングが OFF の場合は、 *sql_chains* のカウントは、 *sql_stmts* のカウントと等しくなります。

使用法 このエレメントを使用すると、処理中に 2、3、 4 などのデータ伝送回数をいくつのステートメントが使用したかについて統計を得ることができます。(1 つのステートメントを処理するには、少なくとも送信と受信の 2 回以上のデータ伝送が必要です。) この統計により、データベース・レベルおよびアプリケーション・レベルでのデータベースやアプリケーションのアクティビティーやネットワーク・トラフィックの状態がより明確になります。

注: *sql_stmts* モニター・エレメントは、 SQL ステートメントのサーバーへの送信が試行される回数を表します。伝送レベルでは、同一カーソル中のすべてのステートメントは単一の SQL ステートメントとしてカウントされます。

sql_req_id SQL ステートメントの要求 ID

SQL ステートメントでの操作の要求 ID。

表 1039. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_stmt	-

使用法 最初のアプリケーションがデータベースに接続した後、データベース・マネージャーが SQL 操作を処理するごとに、この ID が増分します。この値はデータベース全体でユニークであり、ステートメント操作を一意的に識別します。

sql_reqs_since_commit 最終コミット後の SQL 要求

最後のコミット以降にサブミットされた SQL 要求の数。

表 1040. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	基本

使用法 このエレメントを使用すると、トランザクションの進行状況をモニターできます。

sql_stmts 試行された SQL ステートメントの数

データ伝送スナップショットの場合、このエレメントは、ステートメント処理中に DB2 Connect ゲートウェイとホストの間で *n* 回のデータ伝送が行われる際の、SQL ステートメントの数を表します。範囲 *n* は、 *num_transmissions_group* エレメントで指定されます。

表 1041. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS データベース	dcs_dbase	基本
DCS アプリケーション	dcs_appl	基本
データ伝送	stmt_transmissions	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

DCS DATABASE スナップショットの場合、このステートメントのカウントは、データベースが活動化された後のステートメントの数になります。

DCS APPLICATION スナップショットの場合、このステートメントのカウントは、データベースへの接続がこのアプリケーションによって確立された後のステートメントの数になります。

使用法 データベース・レベルまたはアプリケーション・レベルでは、このエレメントを使用してデータベース・アクティビティーを測定します。ある一定の期間について SQL ステートメントのスループットを計算するには、 2 つのスナップショットの間の経過時間の値でこのエレメントの値を割ります。

データ伝送レベルの場合: このエレメントを使用すると、処理中に 2、3、4 などのデータ伝送回数をいくつのステートメントが使用したかについて統計を得ることができます。 (1 つのステートメントを処理するには、少なくとも送信と受信の 2 回以上のデータ伝送が必要です。) この統計により、データベース・レベルおよびアプリケーション・レベルでのデータベースやアプリケーションのアクティビティーやネットワーク・トラフィックの状態がより明確になります。

注:

1. *sql_stmts* モニター・エレメントは、 SQL ステートメントのサーバーへの送信が試行される回数を表します。
   - アプリケーション・レベルおよびデータベース・レベルでは、カーソル中の個々の SQL ステートメントは個別にカウントされます。
   - 伝送レベルでは、同一カーソル中のすべてのステートメントは単一の SQL ステートメントとしてカウントされます。

sqlca SQL 連絡域 (SQLCA)

ステートメントの完了時にアプリケーションに戻された SQLCA データ構造体。

表 1042. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_stmt	-
アクティビティー	event_activity	-

使用法

SQLCA データ構造体を使用すると、ステートメントが正常に終了したかどうかを判別できます。 SQLCA の内容についての詳細は、「*SQL リファレンス 第 1 巻* 」の『SQLCA (SQL 連絡域)』 または「*管理 API リファレンス* 」の『SQLCA データ構造』を参照してください。

sqlrowsread_threshold_id - SQL 読み取り行数しきい値 ID : モニター・エレメント

アクティビティーに適用されていた SQLROWSREAD しきい値の ID。

表 1043. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、 SQLROWSREAD しきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

sqlrowsread_threshold_value - SQL 読み取り行数しきい値 : モニター・エレメント

アクティビティーに適用されていた SQLROWSREAD しきい値の上限。

表 1044. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、SQLROWSREAD しきい値がアクティビティーに適用されている場合、その値を判別します。

sqlrowsread_threshold_violated - SQL 読み取り行数しきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが SQLROWSREAD しきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 1045. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、アクティビティーに適用されていた SQLROWSREAD しきい値にアクティビティーが違反したかどうかを判別します。

sqlrowsreadinsc_threshold_id - サービス・クラス内の SQL 読み取り行数しきい値 ID : モニター・エレメント

アクティビティーに適用されていた SQLROWSREADINSC しきい値の ID。

表 1046. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、 SQLROWSREADINSC しきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

sqlrowsreadinsc_threshold_value - サービス・クラス内の SQL 読み取り行数しきい値 : モニター・エレメント

アクティビティーに適用されていた SQLROWSREADINSC しきい値の上限。

表 1047. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、SQLROWSREADINSC しきい値がアクティビティーに適用されている場合、その値を判別します。

sqlrowsreadinsc_threshold_violated - サービス・クラス内の SQL 読み取り行数しきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが SQLROWSREADINSC しきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 1048. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、アクティビティーに適用されていた SQLROWSREADINSC しきい値にアクティビティーが違反したかどうかを判別します。

sqlrowsreturned_threshold_id - 戻される SQL 読み取り行数しきい値 ID : モニター・エレメント

アクティビティーに適用されていた SQLROWSRETURNED しきい値の ID。

表 1049. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、 SQLROWSRETURNED しきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

sqlrowsreturned_threshold_value - 戻される SQL 読み取り行数しきい値 : モニター・エレメント

アクティビティーに適用されていた SQLROWSRETURNED しきい値の上限。

表 1050. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、SQLROWSRETURNED しきい値がアクティビティーに適用されている場合、その値を判別します。

sqlrowsreturned_threshold_violated - 戻される SQL 読み取り行数しきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが SQLROWSRETURNED しきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 1051. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、アクティビティーに適用されていた SQLROWSRETURNED しきい値にアクティビティーが違反したかどうかを判別します。

sqltempspace_threshold_id - SQL 一時スペースしきい値 ID : モニター・エレメント

アクティビティーに適用されていた SQLTEMPSPACE しきい値の ID。

表 1052. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、 SQLTEMPSPACE しきい値がアクティビティーに適用されていた場合、どのしきい値が適用されていたかを判別します。

sqltempspace_threshold_value - SQL 一時スペースしきい値 : モニター・エレメント

アクティビティーに適用されていた SQLTEMPSPACE しきい値の上限。

表 1053. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、SQLTEMPSPACE しきい値がアクティビティーに適用されている場合、その値を判別します。

sqltempspace_threshold_violated - SQL 一時スペースしきい値の違反 : モニター・エレメント

このモニター・エレメントは、アクティビティーが SQLTEMPSPACE しきい値に違反したことを示す場合に「Yes」を戻します。「No」は、そのアクティビティーがまだしきい値に違反していないことを示します。

表 1054. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

このエレメントを使用して、アクティビティーに適用されていた SQLTEMPSPACE しきい値にアクティビティーが違反したかどうかを判別します。

ss_exec_time サブセクション実行経過時間

サブセクションの実行に要した時間 (秒数)。

表 1055. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	サブセクション	ステートメント

表 1056. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_subsection	-

使用法 サブセクションの進行状況を追跡することができます。

ss_node_number サブセクション・ノード番号

サブセクションが実行されたノード。

表 1057. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	サブセクション	ステートメント

表 1058. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_subsection	-

使用法 各サブセクションとそれが実行されたデータベース・パーティションを関連付けるために使用します。

ss_number サブセクション番号

戻された情報に関連したサブセクションを示します。

表 1059. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	サブセクション	ステートメント

表 1060. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_subsection	-

使用法 この数値は、db2expln を使用して取得可能なアクセス・プラン内のサブセクションに関連しています。

ss_status サブセクションの状況

実行中のサブセクションの現在の状況。

表 1061. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	サブセクション	ステートメント

使用法 現在の状況の値として、次のものがあります。

- 実行中 (sqlmon.h の SQLM_SSEXEC)
- ロック待ち
- 表キュー (table queue) でデータの受信待ち
- 表キュー (table queue) でデータの送信待ち

ss_sys_cpu_time サブセクションに使用されたシステム CPU 時間

現在実行中のステートメント・サブセクションによって使用されたシステム CPU 時間の合計 (秒およびマイクロ秒単位)。

エレメント ID

ss_sys_cpu_time

エレメント・タイプ

時間

表 1062. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	サブセクション	タイム・スタンプ

表 1063. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_subsection	タイム・スタンプ

使用法 このエレメントと CPU 時間に関連する他のエレメントを組み合わせて使用すると、アプリケーション内のアクティビティーのレベルがわかります。また、さらに調整するとその効果が得られる可能性があるアプリケーションを識別できます。

システム CPU は、システム呼び出しに要した時間を示します。ユーザー CPU は、データベース・マネージャーのコードを実行するのに要した時間を示します。

ss_usr_cpu_time サブセクションに使用されたユーザー CPU 時間

現在実行中のステートメント・サブセクションによって使用されたユーザー CPU 時間の合計 (秒およびマイクロ秒単位)。

エレメント ID

ss_usr_cpu_time

エレメント・タイプ

時間

表 1064. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	サブセクション	タイム・スタンプ

表 1065. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_subsection	タイム・スタンプ

使用法 このエレメントと CPU 時間に関連する他のエレメントを組み合わせて使用すると、アプリケーション内のアクティビティーのレベルがわかります。また、さらに調整するとその効果が得られる可能性があるアプリケーションを識別できます。

システム CPU は、システム呼び出しに要した時間を示します。ユーザー CPU は、データベース・マネージャーのコードを実行するのに要した時間を示します。

start_time イベント開始時刻

作業単位開始、ステートメント開始、またはデッドロック検出の日時。このエレメントは、event_start API 構造内ではイベント・モニターの開始を示します。

表 1066. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_start	タイム・スタンプ
トランザクション	event_xact	タイム・スタンプ
ステートメント	event_stmt	タイム・スタンプ
デッドロック	event_deadlock	タイム・スタンプ
デッドロック	event_dlconn	タイム・スタンプ
詳細付きデッドロック	event_detailed_dlconn	タイム・スタンプ

使用法 このエレメントを使用すると、デッドロック接続レコードとデッドロック・イベント・レコードを関連付けることができます。 *stop_time* と組み合わせて使用すると、ステートメントの経過時間またはトランザクション実行時間を計算できます。

注: 「タイム・スタンプ」スイッチが OFF のときは、このエレメントは「0」を報告します。

static_sql_stmts 試行された静的 SQL ステートメント

試行された静的 SQL ステートメントの数。

表 1067. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1068. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 このエレメントを使用すると、データベース・レベルまたはアプリケーション・レベルで成功した SQL ステートメントの合計数を計算できます。

```
  dynamic_sql_stmts
+ static_sql_stmts
- failed_sql_stmts
= モニター期間中のスループット
```

statistics_timestamp 統計タイム・スタンプ : モニター・エレメント

この統計レコードが生成された時刻。

表 1069. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-

表 1069. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats	-
統計	event_wcstats	-
統計	event_qstats	-
統計	event_histogrambin	-

使用法

このエレメントを使用すると、この統計レコードが生成された時点を判別できます。

このエレメントと **last_wlm_reset** エレメントを組み合わせて使用すると、この統計レコードの統計が生成された時間間隔を識別できます。

このモニター・エレメントを使用すると、同じ収集間隔において生成されたすべての統計レコードをグループ化することもできます。

stats_cache_size - 統計キャッシュのサイズ: モニター・エレメント

統計キャッシュの現在のサイズ。統計キャッシュは、リアルタイム統計収集により生成された統計情報をキャッシュに入れるためのカタログ・パーティションで使用されます。

注: 統計キャッシュはカタログ・パーティションにあるため、カタログ・パーティションで取られたスナップショットのみ統計キャッシュ・サイズを報告します。その他のパーティションで取られたスナップショットは、代わりにゼロの値を報告します。グローバル・スナップショットを取る際には、すべてのデータベース・パーティションで報告された値が集約されます。

表 1070. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	-

表 1071. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法

このエレメントを使用すると、現在の統計キャッシュのサイズを判別できます。この値は頻繁に変わります。システム使用量を評価するには、長期にわたり特定のインターバルを設けてスナップショットを取ってください。このエレメントを使用すると、**catalogcache_sz** 構成パラメーターの値を調整できます。

stats_fabricate_time - 統計作成アクティビティーに費やされた合計時間: モニター・エレメント

リアルタイム統計収集により統計作成で費やされた合計時間 (ミリ秒単位)。統計作成とは、照会をコンパイルする際に、統計を生成するのに必要な統計収集アクティビティーのことです。このモニター・エレメントがデータベース・レベルで収集される場合、データベース上で実行中のすべてのアプリケーションに対するリアルタイム統計収集アクティビティーで費やされた合計時間を表します。これがステートメント・レベルで収集される場合、そのステートメントの最新のリアルタイム統計収集アクティビティーで費やされた時間を表します。すべてのデータベース・パーティションで報告された時間は集約されます。

表 1072. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	ステートメント
動的 SQL	dynsql	ステートメント

スナップショット・モニターの場合、このエレメントはリセットできます。

表 1073. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
ステートメント	event_stmt	-

使用法

このエレメントと **stats_fabrications** を組み合わせて使用すると、データベース・レベルのリアルタイム統計収集のパフォーマンスへの影響を評価できます。動的 SQL のスナップショット・モニターの場合、このエレメントと **total_exec_time** および **num_executions** を組み合わせて使用すると、統計作成の影響を評価できます。ステートメント・イベント・モニターの場合、このエレメントを **stmt_start** および **stmt_stop** と結合させて使用すると、リアルタイム統計収集の影響をさらに評価することができます。

stats_fabrications - 統計作成の合計数: モニター・エレメント

すべてのデータベース・アプリケーションに関する照会のコンパイル中にリアルタイム統計により処理される統計作成の合計数。表または索引に保管されているデータをスキャンして統計を取得するのではなく、統計は索引およびデータ・マネージャーによって保守されているメタデータに基づいて作成されます。すべてのデータベース・パーティションで報告された値が集約されます。

表 1074. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1075. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法

このエレメントを使用すると、データベースの統計作成の頻度を判別できます。この値は頻繁に変わります。システム使用量の全体像をより正確に知るには、長期にわたり特定のインターバルを設けてスナップショットを取ってください。このエレメントと **stats_fabricate_time** を組み合わせて使用すると、統計作成の影響を評価する助けになります。

status_change_time アプリケーション状況変更時刻

アプリケーションが現在の状況になった日時。

エレメント ID

status_change_time

エレメント・タイプ

timestamp

表 1076. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_id_info	作業単位、タイム・スタンプ
ロック	appl_lock_list	作業単位、タイム・スタンプ
DCS アプリケーション	dcs_appl_info	作業単位、タイム・スタンプ

使用法 このエレメントを使用して、アプリケーションが現在の状況になっている時間を判別できます。同じ状況が長時間にわたり継続している場合は、問題が起きている可能性があります。

stmt_elapsed_time 最新のステートメント経過時間

最後に完了したステートメントの実行経過時間。

表 1077. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント、タイム・スタンプ
DCS ステートメント	dcs_stmt	ステートメント、タイム・スタンプ

使用法 ステートメントの完了にかかる時間の標識として、このエレメントを使用します。

stmt_first_use_time ステートメントの最初の使用時刻

このエレメントは、ステートメント項目が最初に処理されたときを示します。カーソル操作の場合、**stmt_first_use_time** はカーソルがオープンされたときを示します。アプリケーション調整ノードでは、この値はアプリケーション要求を反映します。非コーディネーター・ノードでは、この値は要求が起点ノードから受信されたときを反映します。

表 1078. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
詳細付きデッドロック履歴値	event_stmt_history	timestamp
詳細付きデッドロック履歴	event_stmt_history	timestamp
アクティビティー	event_activitystmt	timestamp

使用法 このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因となった SQL ステートメントのシーケンスを見ることができます。

stmt_history_id ステートメント履歴 ID

この数値エレメントは、sequence_no エレメントで示された作業単位内でステートメントが実行された位置を、他のステートメント履歴エレメントとの相対位置で示します。作業単位内で最も早く実行されるエレメントは、最も低い値を持ちます。同じ作業単位内で同じステートメントが 2 回実行される場合、2 つの異なる stmt_history_id 値を持つステートメントが 2 箇所示されます。

表 1079. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
詳細付きデッドロック履歴値	event_stmt_history	-
詳細付きデッドロック履歴値	event_data_value	-
詳細付きデッドロック履歴	event_stmt_history	-

使用法 このステートメントを使用して、デッドロックの原因となった SQL ステートメントのシーケンスを見ることができます。

inact_stmthist_sz ステートメント履歴リストのサイズ

履歴付きのデッドロック詳細イベント・モニターが実行している場合、このエレメントは、ステートメント履歴リスト項目を追跡するために、データベース・モニター・ヒープ (MON_HEAP_SZ) から使用されているバイト数を報告します。

表 1080. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	-
database	db	-

使用法 このエレメントは、データベース・モニター・ヒープをチューニングする際に使用できます。

stmt_invocation_id ステートメント呼び出し ID : モニター・エレメント

このエレメントは、SQL ステートメントが実行されたルーチン呼び出しの ID を示します。値は、アプリケーションの中で現在のネスト・レベルがアクティブだった間に、このレベルで発生したルーチン呼び出しの回数を示します。

表 1081. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	event_stmt_history	-
詳細付きデッドロック履歴[1]	event_stmt_history	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

このエレメントを **stmt_nest_level** モニター・エレメントと一緒に使用して、特定の SQL ステートメントの呼び出しを固有に識別できます。また、このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因となった SQL ステートメントのシーケンスを見ることができます。

stmt_isolation ステートメント分離

このエレメントは、ステートメントが実行されていた間にそのステートメントに対して有効だった、分離値を示します。

表 1082. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
詳細付きデッドロック履歴値	event_stmt_history	-
詳細付きデッドロック履歴	event_stmt_history	-
アクティビティー	event_activitystmt	-

考えられる分離レベル値は以下のとおりです。

- SQLM_ISOLATION_LEVEL_NONE 0 (分離レベルが指定されていない)
- SQLM_ISOLATION_LEVEL_UR 1 (非コミット読み取り)
- SQLM_ISOLATION_LEVEL_CS 2 (カーソル固定)
- SQLM_ISOLATION_LEVEL_RS 3 (読み取り固定)
- SQLM_ISOLATION_LEVEL_RR 4 (反復可能読み取り)

使用法 このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因と、特定の SQL ステートメントの実行の動作を理解することができます。

stmt_last_use_time ステートメント最終使用時刻 : モニター・エレメント

このエレメントは、ステートメント項目が最後に処理されたときを示します。カーソル操作の場合、stmt_last_use_time は、カーソルに対して最後のアクションが実行された時刻を示します。そのアクションは、オープン、フェッチ、クローズのいずれかです。アプリケーション調整ノードでは、この値はアプリケーション要求を反映します。非コーディネーター・ノードでは、この値は要求が起点ノードから受信されたときを反映します。

表 1083. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
詳細付きデッドロック履歴値	event_stmt_history	timestamp
詳細付きデッドロック履歴	event_stmt_history	timestamp
アクティビティー	event_activitystmt	-

使用法 このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因となった SQL ステートメントのシーケンスを見ることができます。

stmt_lock_timeout ステートメント・ロック・タイムアウト : モニター・エレメント

このエレメントは、ステートメントが実行されていた間にそのステートメントに対して有効だった、ロック・タイムアウト値を示します。

表 1084. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	event_stmt_history	-
詳細付きデッドロック履歴[1]	event_stmt_history	-
アクティビティー	event_activitystmt	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因と、特定の SQL ステートメントの実行の動作を理解することができます。

stmt_nest_level ステートメント・ネスト・レベル : モニター・エレメント

このエレメントは、ステートメントが実行されていた間にそのステートメントに対して有効だった、ネストまたは再帰のレベルを示します。ネストの各レベルは、ストアード・プロシージャーまたはユーザー定義関数 (UDF) のネストされた呼び出しまたは再帰的呼び出しに対応します。

表 1085. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	event_stmt_history	-
詳細付きデッドロック履歴[1]	event_stmt_history	-
アクティビティー	event_activitystmt	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

このエレメントを **stmt_invocation_id** モニター・エレメントと一緒に使用して、特定の SQL ステートメントの呼び出しを固有に識別できます。また、このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因となった SQL ステートメントのシーケンスを見ることができます。

stmt_node_number ステートメント・ノード

ステートメントが実行されたノード。

表 1086. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント

使用法 各ステートメントをそのステートメントが実行されたノードと関連付けるときに使用します。

stmt_operation/operation ステートメント操作 : モニター・エレメント

現在処理中または (現在実行中のものがない場合は) 最後に処理されたステートメント操作。

表 1087. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント
DCS ステートメント	dcs_stmt	ステートメント

表 1088. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック[1]	event_detailed_dlconn	-
ステートメント	event_stmt	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

このエレメントを使用すると、実行中の操作または最後に終了した操作を判別できます。

次のいずれかになります。

SQL 操作の場合:

- SELECT
- PREPARE
- EXECUTE
- EXECUTE IMMEDIATE
- OPEN
- FETCH
- CLOSE
- DESCRIBE
- STATIC COMMIT
- STATIC ROLLBACK
- FREE LOCATOR
- PREP_COMMIT
- CALL
- PREP_OPEN
- PREP_EXEC
- COMPILE
- DROP PACKAGE

非 SQL 操作の場合:

- RUN STATISTICS
- REORG
- REBIND
- REDISTRIBUTE
- GET TABLE AUTHORIZATION

- GET ADMINISTRATIVE AUTHORIZATION

注: API ユーザーは、データベース・システム・モニターの定数の定義が含まれているヘッダー・ファイル sqlmon.h を参照してください。

stmt_pkgcache_id ステートメント・パッケージ・キャッシュ ID

このエレメントは、動的 SQL ステートメントの内部パッケージ・キャッシュ ID を示します。

表 1089. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	常に収集される

表 1090. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
動的 SQL	dynsql	基本

表 1091. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	event_stmt_history	-
詳細付きデッドロック履歴[1]	event_stmt_history	-
アクティビティー	event_activitystmt	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

複数パーティション環境では、各パーティションに、キャッシュされたステートメントに対する固有のステートメント ID があります。特定のステートメントが、複数のパーティションにわたって同一の ID を持つことはできません。

グローバルな動的 SQL スナップショットでは、最初のステートメント ID のみが戻されます。

stmt_query_id ステートメント照会 ID : モニター・エレメント

このエレメントは、カーソルとして使用された SQL ステートメントに付けられた内部照会 ID を示します。

表 1092. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	event_stmt_history	-
詳細付きデッドロック履歴[1]	event_stmt_history	-
アクティビティー	event_activitystmt	-

使用法

このエレメントを **stmt_nest_level** モニター・エレメントと一緒に使用して、特定の SQL ステートメントの呼び出しを固有に識別できます。また、このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因を理解することもできます。

stmt_sorts ステートメント・ソート回数

stmt_operation を処理するためにデータ集合がソートされた合計回数。

表 1093. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ステートメント
アプリケーション	stmt	ステートメント
動的 SQL	dynsql	ステートメント

使用法 このエレメントを使用すると、索引が必要かどうかを識別できます。索引があればデータをソートする必要性を少なくできるからです。上記の表の関連エレメントを使用すると、このエレメントがソート情報を提供している SQL ステートメントを識別できます。次にこのステートメントを分析し、ソート対象の列を見ると索引候補を判別できます (例えば、 ORDER BY および GROUP BY 節に使用されている列、および結合列)。ソート効率を最適化するために索引が使用されているかどうかを確認する方法については、「*管理ガイド*」の『**EXPLAIN**』を参照してください。

このカウントには、ステートメントを実行するためにデータベース・マネージャーが内部的に生成する一時表のソートが含まれます。ソート数は、SQL ステートメントの最初の FETCH 操作と関連しています。この情報は、ステートメントの操作が最初の FETCH の場合にユーザーに戻されます。ブロック・カーソルの場合は、カーソルが開いたときに複数のフェッチが行われるので注意してください。このような場合、 DB2 が最初の FETCH を内部で発行している間にスナップショットをとる必要があるため、スナップショット・モニターを使用してソート回数を取得するのは困難になります。

ブロック・カーソルを使用して実行されたソートの数を確認するより確実な方法としては、ステートメントに宣言されたイベント・モニターを使用する

方法があります。 CLOSE カーソルのステートメント・イベントにある total_sorts カウンターには、カーソルが定義されたステートメントを実行したときに実行されるソートの合計回数が含まれています。

stmt_source_id ステートメント・ソース ID

このエレメントは、実行された SQL ステートメントのソースに付けられた内部 ID を示します。

表 1094. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	event_stmt_history	-
詳細付きデッドロック履歴[1]	event_stmt_history	-
アクティビティー	event_activitystmt	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

このエレメントを **appl_id** モニター・エレメントと一緒に使用して、特定の SQL ステートメントの実行要求の発信元を固有に識別できます。また、このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因を理解することもできます。

stmt_start ステートメント操作開始タイム・スタンプ

stmt_operation の実行開始日時。

エレメント ID

stmt_start

エレメント・タイプ

timestamp

表 1095. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント、タイム・スタンプ
DCS ステートメント	dcs_stmt	ステートメント、タイム・スタンプ

使用法 このエレメントと stmt_stop を組み合わせて使用すると、ステートメント操作の実行経過時間を計算できます。

stmt_stop ステートメント操作停止タイム・スタンプ

stmt_operation の実行停止日時。

エレメント ID
stmt_stop

エレメント・タイプ
タイム・スタンプ

表 1096. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント、タイム・スタンプ
DCS ステートメント	dcs_stmt	ステートメント、タイム・スタンプ

使用法 このエレメントと stmt_start を組み合わせて使用すると、ステートメント操作の実行経過時間を計算できます。

stmt_sys_cpu_time ステートメントが使用したシステム CPU 時間

現在実行中のステートメントによって使用されたシステム CPU 時間の合計 (秒およびマイクロ秒単位)。

エレメント ID
stmt_sys_cpu_time

エレメント・タイプ
時間

表 1097. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ステートメント、タイム・スタンプ
アプリケーション	stmt	ステートメント、タイム・スタンプ

使用法 このエレメントと CPU 時間に関連する他のエレメントを組み合わせて使用すると、アプリケーション内のアクティビティーのレベルがわかります。また、さらに調整するとその効果が得られる可能性があるアプリケーションを識別できます。

このカウンターには、SQL および非 SQL のステートメントに要した時間のほか、アプリケーションが実行した unfenced ユーザー定義関数 (UDF) およびストアード・プロシージャーも含まれます。

システム CPU は、システム呼び出しに要した時間を示します。ユーザー CPU は、データベース・マネージャーのコードを実行するのに要した時間を示します。

注: ユーザーのオペレーティング・システムでこの情報が利用できないとき、このエレメントは 0 に設定されます。

stmt_text - SQL ステートメント・テキスト : モニター・エレメント

SQL ステートメントのテキスト。

表 1098. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	常に収集される

表 1099. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント
動的 SQL	dynsql	基本
DCS ステートメント	dcs_stmt	ステートメント

表 1100. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック[1]	event_detailed_dlconn	-
詳細付きデッドロック履歴[1]	event_stmt_history	-
ステートメント	event_stmt	-
アクティビティー	event_activitystmt	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

アプリケーション・スナップショットの場合は、このステートメント・テキストに基づいて、スナップショットを取った時点でアプリケーションが何を実行していたかを識別できます。またスナップショットを取った時点でステートメントが処理されていなかった場合は、最後に処理されたものを識別できます。

このエレメントが戻す情報は、SQL ステートメント・キャッシュから取り出されるので、キャッシュがオーバーフローした場合は情報は得られません。ステートメントの SQL テキストを必ずキャプチャーするには、ステートメントのイベント・モニターを使用してください。

動的 SQL ステートメントの場合は、このエレメントを使用してパッケージに関連付けられた SQL テキストを識別します。

ステートメント・イベント・モニターの場合、このエレメントは動的ステートメントについてのみ戻されます。 ステートメント・イベント・モニター・レコードがステートメント・イベント・モニターの BUFFERSIZE オプションで指定されたバッファーのサイズに収まらない場合には、レコードが収まるように **stmt_text** モニターの値が切り捨てられることがあります。

EVENT_STMT_HISTORY イベント・モニターの場合、このエレメントは動的ステートメントについてのみ戻されます。 残されたイベント・モニターの場合、動的ステートメントと静的ステートメントの **stmt_text** は、SQL ステートメント・キャッシュ内で使用可能な場合にのみ戻されます。

パフォーマンスを考慮したために静的 SQL ステートメント・テキストが提供されない場合、これを取得するためにシステム・カタログ表を照会する方法については、**section_number** モニター・エレメントを参照してください。

stmt_type ステートメント・タイプ : モニター・エレメント

処理されるステートメントのタイプ。

表 1101. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	stmt	ステートメント

表 1102. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック[1]	event_detailed_dlconn	-
ステートメント	event_stmt	-
アクティビティー	event_activitystmt	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

このエレメントを使用すると、実行中のステートメントのタイプを判別できます。次のいずれかになります。

- 静的 SQL ステートメント。
- 動的 SQL ステートメント。
- SQL ステートメント以外の操作。例えば、バインドやプリコンパイルなどの操作。

スナップショット・モニターの場合は、このエレメントにより、現在処理中または最後に処理されたステートメントがわかります。

注: API ユーザーは、データベース・システム・モニターの定数の定義が含まれているヘッダー・ファイル sqlmon.h を参照してください。

stmt_usr_cpu_time ステートメントに使用されたユーザー CPU 時間

現在実行中のステートメントによって使用されたユーザー CPU 時間の合計 (秒およびマイクロ秒単位)。

エレメント ID

stmt_usr_cpu_time

エレメント・タイプ

時間

表 1103. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ステートメント、タイム・スタンプ
アプリケーション	stmt	ステートメント、タイム・スタンプ

使用法 このエレメントと CPU 時間に関連する他のエレメントを組み合わせて使用すると、アプリケーション内のアクティビティーのレベルがわかります。また、さらに調整するとその効果が得られる可能性があるアプリケーションを識別できます。

このカウンターには、SQL および非 SQL のステートメントに要した時間のほか、アプリケーションが実行した unfenced ユーザー定義関数 (UDF) およびストアード・プロシージャーも含まれます。

システム CPU は、システム呼び出しに要した時間を示します。ユーザー CPU は、データベース・マネージャーのコードを実行するのに要した時間を示します。

注: ユーザーのオペレーティング・システムでこの情報が利用できないとき、このエレメントは 0 に設定されます。

stmt_value_data 値データ

このエレメントは、SQL ステートメントに対するデータ値のストリング表記です。LOB、LONG および構造化タイプ・パラメーターは空ストリングとして示されます。日付、時刻、およびタイム・スタンプ・フィールドは ISO フォーマットで記録されます。

表 1104. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	stmt_value_data	-
アクティビティー	event_activityvals	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、

将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因を理解することができます。

stmt_value_index 値索引

このエレメントは、SQL ステートメントで使用される入力パラメーター・マーカーまたはホスト変数の位置を表します。

表 1105. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	stmt_value_data	-
アクティビティー	event_activityvals	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因を理解することができます。

stmt_value_isnull NULL 値の値 : モニター・エレメント

このエレメントは、SQL ステートメントに関連したデータ値が NULL 値かどうかを示します。

表 1106. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	stmt_value_isnull	-
アクティビティー	event_activityvals	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因を理解することができます。

stmt_value_isreopt ステートメント再最適化に使用される変数 : モニター・エレメント

このエレメントは、提供された値がステートメント再最適化中に使用された値かどうかを示します。ステートメントが再最適化され (例えば、REOPT BIND オプションの設定のため)、かつこの再最適化中に SQL コンパイラーへの入力として値が使用された場合、値「True」が戻されます。

表 1107. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	event_data_value	-
アクティビティー	event_activityvals	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

このエレメントを提供されたコンパイル環境と一緒に使用して、SQL コンパイラーによる SQL ステートメントの処理を完全に分析できます。

stmt_value_type 値タイプ : モニター・エレメント

このエレメントは、SQL ステートメントに関連したデータ値のタイプのストリング表記です。

表 1108. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
詳細付きデッドロック履歴値[1]	stmt_value_type	-
アクティビティー	event_activityvals	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

このエレメントを他のステートメント履歴項目と一緒に使用して、デッドロックの原因を理解することができます。

sto_path_free_sz - 自動ストレージ・パスのフリー・スペース

このエレメントは、ストレージ・パスが指し示すファイル・システム上で使用可能なフリー・スペースの量を示します。複数のストレージ・パスが同じファイル・システムを指す場合、空きサイズはそれらのストレージ・パス間で分割されません。

エレメント ID
fs_free_size

エレメント・タイプ
情報

表 1109. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	db_sto_path_info	バッファー・プール

使用法 このエレメントを次のエレメントと共に使用して、データベースのスペース使用率に関するノードごとのデータを収集することができます。

- db_storage_path
- fs_used_size
- fs_total_size
- fs_id
- fs_type

stop_time イベント停止時刻

ステートメントが実行を停止した日時。

表 1110. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_stmt	タイム・スタンプ

使用法 このエレメントと *start_time* を組み合わせて使用すると、ステートメントの実行経過時間を計算できます。

FETCH ステートメント・イベントの場合は、最後に正常なフェッチが行われた時刻です。

注: 「タイム・スタンプ」スイッチが OFF のときは、このエレメントは「0」を報告します。

stored_proc_time ストアード・プロシージャー時間

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時点か、またはデータベース・モニター・カウンターの最後のリセット時点以降に、このフェデレーテッド・サーバー・インスタンス上で実行されているすべてのアプリケーションまたは単一アプリケーションからのストアード・プロシージャー・ステートメントに対して、このデータ・ソースが応答に要した合計時間が含まれています (ミリ秒単位)。

表 1111. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	タイム・スタンプ
アプリケーション	appl_remote	タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

応答時間は、フェデレーテッド・サーバーがストアード・プロシージャーをデータ・ソースにサブミットしてからデータ・ソースがストアード・プロシージャーを処理したことを応答するまでの時間です。

使用法 このエレメントを使用すると、このデータ・ソースでストアード・プロシージャーの処理に要した実際の時間を判別できます。

stored_procs ストアード・プロシージャー数

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時点か、またはデータベース・モニター・カウンターの最後のリセット時点以降に、いずれかのアプリケーションに代わってフェデレーテッド・サーバーがこのデータ・ソースで呼び出したストアード・プロシージャーの合計数が含まれています。

表 1112. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、フェデレーテッド・データベースでローカルに行われたストアード・プロシージャーの呼び出し数、またはアプリケーションがフェデレーテッド・データベースに対して呼び出したストアード・プロシージャーの呼び出し数を判別できます。

sync_runstats - 同期 RUNSTATS アクティビティーの合計数: モニター・エレメント

データベース内のすべてのアプリケーションのリアルタイム統計収集により起動される同期 RUNSTATS アクティビティーの合計数。この値には、同期 RUNSTATS コマンドにおいて、成功したものと成功しなかったものの両方が含まれます。すべてのデータベース・パーティションで報告された値が集約されます。

表 1113. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1114. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法

このモニター・エレメントを使用すると、データベースのリアルタイム統計収集により起動された同期 RUNSTATS アクティビティーの数を判別できます。この値は頻繁に変わります。システム使用量をより正確に知るには、長期にわたり特定のインターバルを設けてスナップショットを取ってください。このエレメントと **sync_runstats_time** を組み合わせて使用すると、リアルタイム統計収集により起動された同期 RUNSTATS アクティビティーのパフォーマンスへの影響を評価する助けになります。

sync_runstats_time - 同期 RUNSTATS アクティビティーに費やされた合計時間: モニター・エレメント

リアルタイム統計収集により起動される同期 RUNSTATS アクティビティーの合計時間 (ミリ秒単位)。同期 RUNSTATS アクティビティーは、照会のコンパイル中に発生します。データベース・レベルでは、このモニター・エレメントは、リアルタイム統計収集により起動された、データベース上で実行中のすべてのアプリケーションに対する同期 RUNSTATS アクティビティーで費やされた合計時間を表します。ステートメント・レベルでは、リアルタイム統計収集により起動された、特定のステートメントに対する最新の同期 RUNSTATS アクティビティーで費やされた時間を表します。すべてのデータベース・パーティションで報告された値が集約されます。

表 1115. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	ステートメント
動的 SQL	dynsql	ステートメント

スナップショット・モニターの場合、このエレメントはリセットできます。

表 1116. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
ステートメント	event_stmt	-

使用法

このエレメントと **sync_runstats** を組み合わせて使用すると、データベース・レベルでリアルタイム統計収集により起動された、同期 RUNSTATS アクティビティーのパフォーマンスへの影響を評価できます。

動的 SQL のスナップショット・モニターの場合、このエレメントを **total_exec_time** および **num_executions** と組み合わせて使用すると、同期 RUNSTATS の照会パフォーマンスへの影響を評価できます。

ステートメント・イベント・モニターの場合、このエレメントを **stmt_start** および **stmt_stop** と組み合わせて使用すると、リアルタイム統計収集の影響をさらに評価することができます。

system_cpu_time システム CPU 時間

データベース・マネージャーのエージェント・プロセス、作業単位、またはステートメントで使用されたシステム CPU 時間の合計 (秒およびマイクロ秒単位)。

ステートメント・モニター・スイッチまたはタイム・スタンプ・スイッチがオンになっていない場合は、このエレメントは収集されません。この場合には、このモニター・エレメントは代わりに -1 を表示します。

表 1117. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	-
トランザクション	event_xact	-
ステートメント	event_stmt	-
アクティビティー	event_activity	-

使用法 このエレメントと CPU 時間に関連する他のエレメントを組み合わせて使用すると、アプリケーション内のアクティビティーのレベルがわかります。また、さらに調整するとその効果が得られる可能性があるアプリケーションを識別できます。

注: ユーザーのオペレーティング・システムでこの情報が利用できないとき、このエレメントは 0 に設定されます。

tab_file_id - 表ファイル ID : モニター・エレメント

表のファイル ID (FID)。

表 1118. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLE_METRICS 表関数 - 表メトリックの取得	常に収集される

使用法

tab_type - 表タイプ : モニター・エレメント

このインターフェースは、sqlmon.h 内の定義に基づいたテキスト ID を戻します。USER_TABLE、DROPPED_TABLE、TEMP_TABLE、CATALOG_TABLE、または REORG_TABLE のいずれかになります。

表 1119. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLE_METRICS 表関数 - 表メトリックの取得	常に収集される

使用法

table_file_id - 表ファイル ID : モニター・エレメント

これは表のファイル ID (FID) です。

表 1120. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLE 表関数 - 表メトリックの取得	常に収集される

表 1121. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	ロック
表	table	基本
ロック	appl_lock_list	ロック
ロック	lock	ロック

表 1122. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
デッドロック	lock	-

使用法 このエレメントは情報提供のみを目的としています。データベース・システム・モニターの以前のバージョンとの互換性について情報を戻します。表を個別に識別することはできません。 **table_name** および **table_schema** モニター・エレメントを使用して、表を識別します。

table_name - 表名 : モニター・エレメント

表の名前。

表 1123. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	常に収集される
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

表 1124. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本
アプリケーション	appl	ロック
ロック	appl_lock_list	ロック
ロック	lock	ロック
ロック	lock_wait	ロック

表 1125. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
表	event_table	-
デッドロック[1]	lock	-
デッドロック[1]	event_dlconn	-
詳細付きデッドロック[1]	event_detailed_dlconn	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

このエレメントと **table_schema** を組み合わせて使用すると、リソースの競合の原因を判別できます。

アプリケーション・レベル、アプリケーション・ロック・レベル、およびデッドロック・モニター・レベルでは、現在別のアプリケーションにロックされているためにロック待ちになっているアプリケーションの表を示します。スナップショット・モニターの場合、この項目が有効なのは、「lock」モニター・グループ情報が ON に設定され、さらにアプリケーションが表ロックの取得待ちであることを **lock_object_type** が示している場合に限ります。

オブジェクト・レベルのスナップショット・モニターの場合は、表レベルおよび行レベルのロックについてこの項目が戻されます。このレベルで報告される表は、このアプリケーションがこれらのロックを保留している表です。

表レベルのスナップショット・モニターおよびイベント・モニターの場合は、情報が収集された表を示します。一時表の場合、 **table_name** の形式は 『TEMP (*n*, *m*)』 です。

- *n* は表スペース ID
- *m* は **table_file_id** エレメント

table_scans - 表スキャン : モニター・エレメント

この表でのスキャンの数。

表 1126. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	常に収集される

使用法

table_schema - 表スキーマ名 : モニター・エレメント

表のスキーマ。

表 1127. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	常に収集される
MON_GET_INDEX 表関数 - 索引メトリックの取得	常に収集される

表 1128. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本
アプリケーション	appl	ロック
ロック	appl_lock_list	ロック
ロック	lock	ロック
ロック	lock_wait	ロック

表 1129. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
表	event_table	-
デッドロック[1]	lock	-
デッドロック[1]	event_dlconn	-
詳細付きデッドロック[1]	event_detailed_dlconn	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、

将来のリリースではサポートされなくなる予定です。 CREATE EVENT MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

このエレメントと **table_name** を組み合わせて使用すると、リソースの競合の原因を判別できます。

アプリケーション・レベル、アプリケーション・ロック・レベル、およびデッドロック・モニター・レベルでは、現在別のアプリケーションにロックされているためにロック待ちになっているアプリケーションの表のスキーマを示します。このエレメントは、アプリケーションが表ロックの取得待ちであることを **lock_object_type** が示している場合にのみ設定できます。アプリケーション・レベルおよびアプリケーション・ロック・レベルでのスナップショット・モニターの場合、この項目は「lock」モニター・グループ情報が ON に設定されている場合にのみ有効です。

オブジェクト・レベルのスナップショット・モニターの場合は、表レベルおよび行レベルのロックについてこの項目が戻されます。このレベルで報告される表は、このアプリケーションがこれらのロックを保留している表です。

表レベルのスナップショット・モニターおよびイベント・モニターの場合は、このエレメントは情報が収集された表スキーマを示します。一時表の場合、**table_schema** の形式は 『*<agent_id><auth_id>*』 です。

- *agent_id* は、一時表を作成しているアプリケーションのアプリケーション・ハンドルです。
- *auth_id* は、アプリケーションがデータベースに接続するときに使用する許可 ID です。

table_type - 表タイプ : モニター・エレメント

情報が戻される表のタイプ。

表 1130. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_TABLE 表関数 - 表メトリックの取得	常に収集される

表 1131. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本

表 1132. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表	event_table	-

使用法

このエレメントを使用すると、情報が戻される表の識別に役立ちます。表がユーザー表またはシステム・カタログ表の場合は、 **table_name** および **table_schema** を使用して表を識別できます。

表のタイプは、次のいずれかになります。可能な値は、sqlmon.h ファイル内の定義に基づくテキスト・ストリングです。

USER_TABLE
ユーザー表。

TEMP_TABLE
一時表。表が使用された後に表がデータベース内に保持されない場合も、一時表に関する情報が戻されます。その場合でも、このタイプの表に関する情報は役に立ちます。

CATALOG_TABLE
システム・カタログ表。

tablespace_auto_resize_enabled - 表スペースの自動サイズ変更可能 : モニター・エレメント

このエレメントは、表スペースの自動サイズ変更が使用可能かどうかを記述します。値 1 は「はい」を意味し、値 0 は「いいえ」を意味します。

表 1133. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1134. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本

使用法

このエレメントは、DMS 表スペースおよび非一時自動ストレージ表スペースにのみ適用されます。このエレメントが 1 に設定されている場合、自動サイズ変更は使用可能です。表スペースの増加率および最大サイズについては、次のモニター・エレメントを参照してください。

- **tablespace_max_size**
- **tablespace_increase_size**
- **tablespace_increase_size_percent**

tablespace_content_type - 表スペースのコンテンツ・タイプ : モニター・エレメント

表スペース内のコンテンツのタイプ。

表 1135. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1136. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本

使用法

表スペース内のコンテンツのタイプ (sqlmon.h 内に定義) は、次のいずれかになります。

- すべてのタイプの永続データ。
  - REGULAR 表スペース: SQLM_TABLESPACE_CONTENT_ANY
  - LARGE 表スペース: SQLM_TABLESPACE_CONTENT_LARGE
- システム一時データ: SQLM_TABLESPACE_CONTENT_SYSTEMP
- ユーザー一時データ: SQLM_TABLESPACE_CONTENT_USRTEMP

tablespace_cur_pool_id - 現在使用中のバッファー・プール : モニター・エレメント

表スペースが現在使用中のバッファー・プールのバッファー・プール ID。

表 1137. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1138. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本

使用法 バッファー・プールは、それぞれ固有の整数で識別できます。このエレメントの値は、SYSCAT.BUFFERPOOLS ビューの BUFFERPOOLID 列の値と一致します。

tablespace_current_size 表スペースの現行サイズ

このエレメントは、表スペースの現行サイズをバイト数で示します。

表 1139. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 DMS および自動ストレージ表スペースの場合、このエレメントはすべての表スペース・コンテナーの合計サイズをバイト数で表します。この値は、表スペースの合計ページ (tablespace_total_pages) に表スペースのページ・サイズ (tablespace_page_size) を掛けた値に等しくなります。このエレメントは、SMS 表スペース、または一時自動ストレージ表スペースには適用されません。

自動ストレージ表スペースの表スペース作成時に、現行サイズが初期サイズと一致しないことがあります。現行サイズの値は、作成時の初期サイズのページ・サイズ、エクステント・サイズ、およびストレージ・パスの数をすべて掛け合わせた値の範囲内になります (通常は大きくなりますが、場合によっては小さくなることもあります)。常に tablespace_max_size 以下になります (設定されている場合)。これは、コンテナーがフル・エクステント単位でしか大きくなれず、かつ複数のコンテナーがセットとして大きくならなければならないためです。

tablespace_extent_size - 表スペースのエクステント・サイズ : モニター・エレメント

表スペースが使用するエクステント・サイズ。

表 1140. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1141. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本

tablespace_free_pages 表スペース内のフリー・ページ数 : モニター・エレメント

1 つの表スペース内で現在フリーの合計ページ数。

表 1142. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1143. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法

これは、DMS 表スペースにのみ適用できます。

tablespace_id - 表スペース ID : モニター・エレメント

現行データベースが使用する表スペースを一意的に示す整数。

表 1144. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLE 表関数 - 表メトリックの取得	常に収集される
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_EXTENT_MOVEMENT_STATUS - エクステントの移動の進行状況メトリックの取得	常に収集される

表 1145. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本
表	table	基本

表 1146. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表	event_table	-

使用法

このエレメントの値は、SYSCAT.TABLESPACES のビューの TBSPACEID 列の値と一致します。

tablespace_increase_size バイト単位のサイズの増加

このエレメントは、自動サイズ変更表スペースがいっぱいになって、さらにスペースが必要になったときに、表スペースをどれだけ大きくするかをバイト数で示します。

表 1147. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 これは、自動サイズ変更可能な表スペースがいっぱいになって、さらにスペースが必要になり、かつ表スペースの最大サイズに達していない場合に、表スペースに追加するスペースの量を示します。このエレメントの値が -1 (またはスナップショット出力で 『AUTOMATIC』) の場合、スペースの追加が必要になったときに DB2 が自動的に値を決定します。このエレメントは、自動サイズ変更が使用可能になっている表スペースにのみ適用されます。

tablespace_increase_size_percent パーセント単位のサイズの増加 : モニター・エレメント

このエレメントは、自動サイズ変更表スペースがいっぱいになって、さらにスペースが必要になったときに、表スペースをどれだけ大きくするかを示します。実際のバイト数は、表スペースがサイズ変更されるときに、そのときの表スペースのサイズに基づいて決まります。

表 1148. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 これは、自動サイズ変更可能な表スペースがいっぱいになって、さらにスペースが必要になり、かつ表スペースの最大サイズに達していない場合に、表スペースに追加するスペースの量を示します。増加率は、表スペースがサイズ変更されるときの、表スペースの現行サイズ (tablespace_current_size) のパーセンテージに基づいています。このエレメントは、自動サイズ変更が使用可能になっている表スペースにのみ適用されます。

tablespace_initial_size 表スペースの初期サイズ

自動ストレージ表スペースの初期サイズ (バイト数)。

表 1149. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 非一時自動ストレージ表スペースの場合、このモニター・エレメントは、表スペースが作成されたときのその表スペースの初期サイズをバイト数で表します。

tablespace_last_resize_failed 失敗した最後のサイズ変更

このエレメントは、表スペースのサイズを自動的に大きくする最後の試みが失敗したかどうかを記述します。値 1 は「はい」を意味し、0 は「いいえ」を意味します。

表 1150. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 自動ストレージ表スペースの場合、このエレメントは、データベースのどのストレージ・パスにもスペースが残っていないことを示している可能性があります。非自動ストレージ表スペースの場合、失敗とは、コンテナーのファイル・システムがいっぱいだったため、コンテナーのいずれかが拡張できなかったことを意味します。失敗の別の理由は、表スペースの最大サイズに達していることです。このエレメントは、自動サイズ変更が使用可能になっている表スペースにのみ適用されます。

tablespace_last_resize_time 最後にサイズ変更が正常に行われた時刻

このエレメントは、表スペースのサイズが正常に大きくなった最後の時刻を表すタイム・スタンプを示します。

表 1151. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 自動的にサイズ変更可能な表スペースの場合、このエレメントは、その表スペースがいっぱいになって、さらにスペースが必要になり、かつ表スペースの最大サイズに達しておらず、表スペースに自動的にスペースが追加された最後の時刻を表します。このエレメントは、自動サイズ変更が使用可能になっている表スペースにのみ適用されます。

tablespace_max_size 表スペースの最大サイズ

このエレメントは、表スペースが自動的にサイズ変更したり、大きくなったりできる最大サイズをバイト数で示します。

表 1152. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 これは、自動的にサイズ変更可能な表スペースが、どの最大サイズまで自動的に大きくなれるかをバイト数で示します。この値が tablespace_current_size エレメントと等しい場合、表スペースが大きくなる余地はありません。このエレメントの値が -1 の場合、最大サイズは「無制限」と見なされ、ファイル・システムがフルになるか、表スペースの体系的なサイズ制限に達するまで、表スペースは自動的にサイズ変更できます (この制限については、「*SQL リファレンス*」の『SQL Limits』の付録で説明されています)。このエレメントは、自動サイズ変更が使用可能になっている表スペースにのみ適用されます。

tablespace_min_recovery_time ロールフォワードの最小リカバリー時間

表スペースをロールフォワードできる最も早い時点を示すタイム・スタンプ。

エレメント ID

tablespace_min_recovery_time

エレメント・タイプ

情報

表 1153. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 ゼロ以外のときだけ表示されます。

tablespace_name - 表スペース名 : モニター・エレメント

表スペースの名前。

表 1154. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_EXTENT_MOVEMENT_STATUS - エクステントの移動の進行状況メトリックの取得	常に収集される

表 1155. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本
ロック	appl_lock_list	基本
ロック	lock	ロック
ロック	lock_wait	ロック

表 1156. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
デッドロック[1]	lock	-
デッドロック[1]	event_dlconn	-
詳細付きデッドロック[1]	event_detailed_dlconn	-
表スペース	tablespace_list	-

1 このオプションは推奨されなくなりました。この使用は推奨されておらず、将来のリリースではサポートされなくなる予定です。 CREATE EVENT

MONITOR FOR LOCKING ステートメントを使用して、ロック・タイムアウト、ロック待機、およびデッドロックなど、ロック関連のイベントをモニターしてください。

使用法

このエレメントは、リソースの競合の原因を判別するときに役立ちます。

これはデータベース・カタログ表の SYSCAT.TABLESPACES にある TBSPACE 列と同じです。アプリケーション・レベル、アプリケーション・ロック・レベル、およびデッドロック・モニター・レベルでは、アプリケーションがロックを待機している表スペースの名前です。ほかのアプリケーションがこの表スペースのロックを保留しています。

ロック・レベルでは、アプリケーションがロックを保留している表スペースの名前です。

表スペース・レベルでは (バッファー・プール・モニター・グループが ON の場合)、情報が戻される表スペースの名前です。

このエレメントは、パーティション表で保持されている表ロックの場合は戻されません。

tablespace_next_pool_id - 次の始動時に使用されるバッファー・プール : モニター・エレメント

データベースを次に始動したときに表スペースが使用するバッファー・プールのバッファー・プール ID。

表 1157. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1158. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本

使用法 バッファー・プールは、それぞれ固有の整数で識別できます。このエレメントの値は、SYSCAT.BUFFERPOOLS ビューの BUFFERPOOLID 列の値と一致します。

tablespace_num_containers 表スペース内のコンテナー数

表スペース内のコンテナーの合計数。

表 1159. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

tablespace_num_quiescers - 静止プログラム数

表スペースを静止させているユーザーの数 (0 から 5 の範囲)。

表 1160. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 この値は表スペースを静止させたエージェントの数を示します (「共有」、「更新」、または「排他」モード)。それぞれの静止プログラムについて、次の情報が tablespace_quiescer 論理データ・グループに戻されます。

- 静止プログラムのユーザー許可 ID
- 静止プログラムのエージェント ID
- この表スペースが静止することになった、静止されたオブジェクトの表スペース ID
- この表スペースが静止することになった、静止されたオブジェクトのオブジェクト ID
- 静止状態

tablespace_num_ranges 表スペース・マップ内の範囲数

表スペース・マップ内の範囲 (項目) の数。この範囲は 1 から 100 です (ただし通常は 12 未満)。表スペース・マップがあるのは、DMS 表スペースの場合だけです。

表 1161. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

tablespace_page_size - 表スペースのページ・サイズ : モニター・エレメント

表スペースが使用するページ・サイズ (バイト単位)。

表 1162. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1163. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本

tablespace_page_top 表スペース最高水準点 : モニター・エレメント

最高水準点を保持している表スペース内のページ。

表 1164. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1165. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法

DMS の場合、このエレメントは、表スペースで最後に割り振られたエクステントの次にある最初のフリー・エクステントのページ番号を示します。この値は減少するので、実際の「最高水準点」ではなく「現在の水準点」であることに注意してください。この情報は SMS には適用できません。

tablespace_paths_dropped - ドロップされたパスを使用している表スペース : モニター・エレメント

ドロップされたストレージ・パスを表スペースが使用していることを示します。

表 1166. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1167. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法

自動ストレージを使用する表スペースの場合に、このモニター・エレメントを使用して、ドロップされたストレージ・パスに表スペースのコンテナーがあるかどうかを判別します。ストレージ・パスがデータベースから物理的にドロップされる前に、すべての表スペースはその使用を停止しなければなりません。ドロップされたストレージ・パスの使用を停止するには、表スペースをドロップするか、または

ALTER TABLESPACE ステートメントの REBALANCE 節を使用して表スペースのリバランスを行います。

tablespace_pending_free_pages 表スペース内のペンディング・フリー・ページ数 : モニター・エレメント

すべてのペンディング・トランザクションがコミットまたはロールバックされ、オブジェクトのための新しいスペースが要求されたときにフリーになる表スペースのページ数。

表 1168. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1169. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法

これは、DMS 表スペースにのみ適用できます。

tablespace_prefetch_size - 表スペースのプリフェッチ・サイズ : モニター・エレメント

プリフェッチャーがディスクから 1 回に取得できる最大ページ数。

表 1170. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1171. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本
表スペース	tablespace_nodeinfo	基本

使用法

- 自動プリフェッチ・サイズが使用可能な場合は、このエレメントは値「-1」を *tablespace* 論理データ・グループ中に報告し、実際の値を *tablespace_nodeinfo* 論理データ・グループ中に報告します。
- 自動プリフェッチ・サイズが使用可能でない場合は、このエレメントは実際の値を *tablespace* 論理データ・グループ中に報告し、 *tablespace_nodeinfo* 論理データ・グループ中には表示されません。

tablespace_rebalancer_extents_processed リバランサーで処理されたエクステントの数

リバランサーの開始後または再始動後 (どちらか後で実行された方) に、リバランサーですでに移動されたエクステントの数。

エレメント ID

tablespace_rebalancer_extents_processed

エレメント・タイプ

情報

表 1172. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 リバランサーの完了レベルを示す標識として使用できます。このエレメントの内容が時間とともに変化する様子を追跡すると、リバランスの進行状況をモニターできます。 tablespace_state と rebalance_mode を使用すると、リバランスが完了したかどうかチェックできます。これは、DMS 表スペースにのみ適用できます。

tablespace_rebalancer_extents_remaining リバランサーで処理されるエクステントの合計数

移動するエクステントの数。この値は、リバランサーの開始時刻または再始動時刻 (どちらか後に実行された方) の時点で計算されます。

エレメント ID

tablespace_rebalancer_extents_remaining

エレメント・タイプ

情報

表 1173. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 リバランサーの完了レベルを示す標識として使用できます。このエレメントの内容が時間とともに変化する様子を追跡すると、リバランスの進行状況をモニターできます。再平衡化が終了したかどうかを確認するには、tablespace_state を使用します。これは、DMS 表スペースにのみ適用できます。

tablespace_rebalancer_last_extent_moved リバランサーによって最後に移動されたエクステント

リバランサーによって最後に移動されたエクステント。

エレメント ID

tablespace_rebalancer_last_extent_moved

エレメント・タイプ
情報

表 1174. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 リバランサーの完了レベルを示す標識として使用できます。このエレメントの内容が時間とともに変化する様子を追跡すると、リバランスの進行状況をモニターできます。 tablespace_state と rebalance_mode を使用すると、リバランスが完了したかどうかチェックできます。これは、DMS 表スペースにのみ適用できます。

tablespace_rebalancer_mode - リバランサー・モード : モニター・エレメント

現在のリバランス処理が表スペースからスペースを除去しているのか、あるいは表スペースにスペースを追加しているのかを示します。

表 1175. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1176. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法

フォワード・リバランス は、新規コンテナーが追加されたとき、または既存のコンテナーのサイズが大きくなったときに行われます。フォワード・リバランス操作では、データ移動は表スペースの最初のエクステントから始まり、最高水準点のエクステントで終わります。

リバース・リバランス は、コンテナーが除去されたりサイズが縮小されたりする場合に、解放されるスペースからデータを移動する必要があるときに行われます。リバース・リバランス操作では、データ移動は最高水準点のエクステントから始まり、表スペースを逆順に移動していき、表スペースの最初のエクステントで終わります。

2 パス・リバランス は、フォワード・リバランスの後にリバース・リバランスが行われる処理です。リバランス操作の一環としてコンテナーの追加とドロップの両方が行われるときに、2 パス・リバランスが行われることがあります。

DMS 非自動ストレージ表スペースの場合、このモニター・エレメントは、表スペースに対して行われているリバランスのタイプを示します。 DMS 非自動表スペースでは、単一のフォワード・リバランスまたは単一のリバース・リバランスのみを行えます。

自動ストレージ表スペースの場合、このモニター・エレメントは、現在のリバランス処理が表スペースに対して何を行っているかを示します。一般に、リバランスが開始されると、必要なのは単一のフォワード・リバランスまたは単一のリバース・リバランスのみです。ただし、自動ストレージ表スペースの場合は、2 パス・リバランスが必要な場合があります。

tablespace_rebalancer_mode に可能な値は、sqlmon.h ファイルで次のように定義されています。

SQLM_TABLESPACE_NO_REBAL
リバランスは行われていません。

SQLM_TABLESPACE_FWD_REBAL
フォワード・リバランスが行われていす。

SQLM_TABLESPACE_REV_REBAL
リバース・リバランスが行われています。

SQLM_TABLESPACE_FWD_REBAL_OF_2PASS
2 パス・リバランスのうちのフォワード・リバランス・フェーズが行われています。

SQLM_TABLESPACE_REV_REBAL_OF_2PASS
2 パス・リバランスのうちのリバース・リバランス・フェーズが行われています。

tablespace_rebalancer_priority 現行のリバランサー優先順位

データベース内で実行されているリバランサーの優先順位。

表 1177. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 これは、DMS 表スペースにのみ適用できます。

tablespace_rebalancer_restart_time リバランサー再始動時刻

リバランサーが休止または停止後に再始動されたときを示すタイム・スタンプ。

エレメント ID
tablespace_rebalancer_restart_time

エレメント・タイプ
情報

表 1178. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 リバランサーの完了レベルを示す標識として使用できます。リバランサーが再始動されたときを示し、リバランサーの速度を導出できるので、推定完了時刻を得られます。これは、DMS 表スペースにのみ適用できます。

tablespace_rebalancer_start_time リバランサー開始時刻

リバランサーを最初に開始した時刻を示すタイム・スタンプ。

エレメント ID

tablespace_rebalancer_start_time

エレメント・タイプ

情報

表 1179. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 リバランサーを最初に開始した時刻を知るのに使用します。これは、リバランサーの処理速度を測定したり、リバランサーの終了時刻を推定するときに使用できます。これは、DMS 表スペースにのみ適用できます。

tablespace_state 表スペースの状態 : モニター・エレメント

このエレメントは、表スペースの現在の状態を示します。

表 1180. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1181. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法

このエレメントには、表スペースの現在の状態を示す 16 進値が含まれています。外部から見ることができる表スペースの状態は、特定の状態値が 16 進数の合計値で構成されています。例えば、状態が ″quiesced: EXCLUSIVE″ かつ ″Load pending″ の場合、その値は 0x0004 + 0x0008、つまり 0x000c となります。 db2tbst コマンドを使用して、特定の 16 進値と関連した表スペース状態を取得します。

表 1182. sqlutil.h 中にリストされているビット定義

16 進値	10 進値	状態
0x0	0	標準 (sqlutil.h 内の SQLB_NORMAL の定義を参照)
0x1	1	静止モードでの共有
0x2	2	静止モードでの更新
0x4	4	静止モードでの排他
0x8	8	ロード・ペンディング
0x10	16	削除ペンディング
0x20	32	バックアップ・ペンディング
0x40	64	ロールフォワード進行中
0x80	128	ロールフォワード・ペンディング
0x100	256	リストア・ペンディング
0x100	256	リカバリー・ペンディング (未使用)
0x200	512	使用不可ペンディング
0x400	1024	REORG 進行中
0x800	2048	バックアップ進行中
0x1000	4096	ストレージを定義する必要がある
0x2000	8192	リストア進行中
0x4000	16384	オフラインのためアクセス不可
0x8000	32768	ドロップ・ペンディング
0x80000	524288	移動進行中
0x2000000	33554432	ストレージを定義可能
0x4000000	67108864	ストレージ定義は最終状態
0x8000000	134217728	ストレージ定義はロールフォワードの前に変更された
0x10000000	268435456	DMS リバランサー進行中
0x20000000	536870912	表スペース削除進行中
0x40000000	1073741824	表スペース作成進行中

tablespace_state_change_object_id 状態変更オブジェクト ID

表スペースの状態を「ロード・ペンディング」または「削除ペンディング」に設定したオブジェクト。

エレメント ID

tablespace_state_change_object_id

エレメント・タイプ

情報

表 1183. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 このエレメントに意味があるのは、表スペースの状態が「ロード・ペンディング」または「削除ペンディング」の場合だけです。このエレメントの値がゼロ以外の場合は、SYSCAT.TABLES ビューの TABLEID 列の値と一致します。

tablespace_state_change_ts_id 状態変更表スペース ID

表スペースの状態が「ロード・ペンディング」または「削除ペンディング」の場合は、表スペースの状態の設定を引き起こしたオブジェクトの表スペース ID が示されます。

エレメント ID

tablespace_state_change_ts_id

エレメント・タイプ

情報

表 1184. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本

使用法 このエレメントに意味があるのは、表スペースの状態が「ロード・ペンディング」または「削除ペンディング」の場合だけです。このエレメントの値がゼロ以外の場合は、 SYSCAT.TABLES ビューの TABLESPACEID 列の値と一致します。

tablespace_total_pages 表スペース内の合計ページ数 : モニター・エレメント

1 つの表スペース内の合計ページ数。

表 1185. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1186. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本 (DMS 表スペース) バッファー・プール (SMS 表スペース)

使用法

1 つの表スペースが使用するオペレーティング・システムの合計スペースです。DMS の場合は、コンテナー・サイズの合計となります (オーバーヘッドを含む)。SMS の場合は、この表スペースに保管される表に使用されるすべてのファイル・ス

ページの合計です (この情報は、バッファー・プール・スイッチが ON の場合にのみ収集されます)。

tablespace_type - 表スペース・タイプ : モニター・エレメント

表スペースのタイプ。

表 1187. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1188. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本

使用法

このエレメントは、この表スペースがデータベース管理の表スペース (DMS) か、システム管理の表スペース (SMS) かを示します。

tablespace_type の値 (sqlmon.h 内に定義) は次のとおりです。

- DMS の場合: SQLM_TABLESPACE_TYP_DMS
- SMS の場合: SQLM_TABLESPACE_TYP_SMS

tablespace_usable_pages 表スペース内の使用可能ページ数 : モニター・エレメント

表スペース内の合計ページ数からオーバーヘッド・ページ数を引いた数値。

表 1189. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1190. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本 (DMS 表スペース) バッファー・プール (SMS 表スペース)

使用法

このエレメントが適用されるのは DMS 表スペースだけです。 SMS 表スペースの場合は、このエレメントの値は **tablespace_total_pages** モニター・エレメントと同じになります。

表スペースのリバランス中に、使用可能ページ数に新たに追加されたコンテナーが加えられますが、これらの新しいページ数は、リバランス完了までの間フリー・ページ数には反映されません。表スペースのリバランスが実行されていない場合は、使用済みページ数、フリー・ページ数、およびペンディング・フリー・ページ数の合計が使用可能ページ数に等しくなります。

tablespace_used_pages 表スペース内の使用されているページ数 : モニター・エレメント

表スペース内で現在使用中 (フリーではない) の合計ページ数。

表 1191. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1192. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace_nodeinfo	基本 (DMS 表スペース) バッファー・プール (SMS 表スペース)

使用法

DMS 表スペースで使用中の合計ページ数です。 SMS 表スペースの場合は、**tablespace_total_pages** モニター・エレメントと同じ値になります。

tablespace_using_auto_storage - 自動ストレージが使用可能な表スペース : モニター・エレメント

このエレメントは、表スペースが自動ストレージ表スペースとして作成されたかどうかを記述します。値 1 は「はい」を意味し、値 0 は「いいえ」を意味します。

表 1193. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1194. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	tablespace	基本

使用法

このエレメントを使用して、指定された表スペースが、明示的に提供されているコンテナーを使用するのではなく、自動ストレージを使用して作成されたかどうか (つまり MANAGED BY AUTOMATIC STORAGE 節を指定して作成されているかどうか) を判別できます。表スペースは、データベースに関連している一部の (または全部の) ストレージ・パスに存在するコンテナーを持つことができます。

tbsp_max_page_top - 最大表スペース・ページの最高水準点 : モニター・エレメント

データベースが活動化された以降に DMS 表スペースに割り振られた最高ページ番号。

表 1195. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

使用法

この値は、**tablespace_page_top** モニター・エレメントの値が増えるときにはいつでも変更されます。

tcpip_recv_volume - TCP/IP 受信ボリューム : モニター・エレメント

データ・サーバーがクライアントから TCP/IP 経由で受信したデータの量。この値はバイト単位で示されます。

表 1196. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1196. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1197. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

tcpip_recv_wait_time - TCP/IP 受信待機時間 : モニター・エレメント

TCP/IP 経由での着信クライアント要求の待機にかかった時間。アイドル時間は除きます。値はミリ秒単位で示されます。

表 1198. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1198. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1199. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

tcpip_recvs_total - TCP/IP 受信の合計 : モニター・エレメント

データベース・サーバーがクライアント・アプリケーションから TCP/IP 経由でデータを受信した回数。

表 1200. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1201. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

tcpip_send_volume - TCP/IP 送信ボリューム : モニター・エレメント

データ・サーバーによってクライアントに送信されたデータの量。この値はバイト単位で示されます。

表 1202. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1203. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

tcpip_send_wait_time - TCP/IP 送信待機時間 : モニター・エレメント

クライアントへの TCP/IP 送信のブロックにかかった時間。値はミリ秒単位で示されます。

表 1204. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1205. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

tcpip_sends_total - TCP/IP 送信の合計 : モニター・エレメント

データベース・サーバーからクライアント・アプリケーションへ TCP/IP 経由でデータが送信された回数。

表 1206. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE

表 1206. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1207. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-

temp_tablespace_top TEMPORARY 表スペースの最上位 : モニター・エレメント

サービス・クラスまたは作業クラスでの、すべてのネスト・レベルにおける DML アクティビティーの TEMPORARY 表スペース使用量の最高水準点 (KB 単位)。サービス・クラスでは、サービス・クラスの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。作業クラスでは、その作業クラスに COLLECT AGGREGATE ACTIVITY DATA 作業アクションが指定されていない場合、このモニター・エレメントは -1 を返します。 ワークロードでは、ワークロードの COLLECT AGGREGATE ACTIVITY DATA が NONE に設定されている場合、このモニター・エレメントは -1 を返します。

サービス・クラスの場合、REMAP ACTIVITY アクションを使用してサービス・サブクラス間のアクティビティーを再マップすると、アクティビティーが完了したサ

ービス・サブクラスの temp_tablespace_top 最高水準点のみが変更されます。 アクティビティーがマップされたサービス・サブクラスでも、アクティビティーがそこで完了しなかった場合、そのサービス・サブクラスの最高水準点は何も影響を受けません。

表 1208. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats	-
統計	event_wcstats	-
統計	event_wlstats	-

使用法

このエレメントを使用して、収集された時間間隔にサービス・クラス、ワークロード、または作業クラス用のパーティションで到達した DML アクティビティーの SYSTEM TEMPORARY 表スペース使用量の最大値を判別することができます。

このエレメントは、適用される TEMPORARY 表スペースのしきい値があるアクティビティーによってのみ更新されます。アクティビティーに TEMPORARY 表スペースのしきい値が適用されない場合、0 の値が返されます。

territory_code データベース・テリトリー・コード

モニター・データが収集されるデータベースのテリトリー・コード。このモニター・エレメントは以前は country_code と呼んでいました。

表 1209. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
アプリケーション	appl	基本

表 1210. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
イベント・ログ・ヘッダー	event_log_header	-
接続	event_connheader	-

使用法 テリトリー・コード情報は、データベース構成ファイルに記録されています。

DRDA AS 接続の場合、このエレメントは 0 に設定されます。

threshold_action しきい値アクション : モニター・エレメント

このしきい値違反レコードが適用されるしきい値のアクション。可能な値は「停止」、「続行」および「再マップ」です。

表 1211. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-

使用法

このエレメントを使用すると、しきい値を違反したアクティビティーが、違反が起きた時点で停止したか、実行を継続できたか、それとも他のサービス・サブクラスに再マップされたかを判別できます。 アクティビティーが停止された場合は、そのアクティビティーをサブミットしたアプリケーションは SQL4712N エラーを受け取ることになります。アクティビティーが他のサービス・サブクラスに再マップされた場合、パーティション上でアクティビティー用に作動しているエージェントはしきい値のターゲット・サービス・サブクラスに移動します。

threshold_domain しきい値ドメイン : モニター・エレメント

このキューに関係するしきい値のドメイン。

可能な値は以下のとおりです。

- データベース
- 作業アクションセット
- サービス・スーパークラス
- サービス・サブクラス
- ワークロード

表 1212. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	-

使用法

このエレメントを使用すると、述部は同じでもドメインが異なるしきい値のキュー統計を区別することができます。

threshold_maxvalue しきい値最大値 : モニター・エレメント

このモニター・エレメントは、キューイング非対象しきい値においては、このしきい値を超えてしまった値を表します。キューのしきい値の場合は、このモニター・エレメントは、キューイングの原因となった並行性のレベルを表します。キューのしきい値の違反の原因となった並行性のレベルは、**threshold_maxvalue** および **threshold_queuesize** モニター・エレメントの合計です。

表 1213. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-

使用法

アクティビティーしきい値では、このエレメントは、しきい値の違反が発生した時点でのしきい値の最大値の履歴レコードを提供します。これは、違反の発生以降にしきい値の最大値が変更され、古い値が SYSCAT.THRESHOLDS ビューで表示できなくなった場合に便利です。

threshold_name しきい値名 : モニター・エレメント

このキューに関係するしきい値の固有の名前。

表 1214. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	-

使用法

このエレメントを使用すると、このレコードが示す統計の元となるキューのしきい値を一意的に識別できます。

threshold_predicate しきい値述部 : モニター・エレメント

違反したしきい値または統計の収集の対象となったしきい値のタイプを識別します。

表 1215. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-
統計	event_qstats	-

使用法

このモニター・エレメントを他の統計またはしきい値違反モニター・エレメントと一緒に使用すると、しきい値違反の分析をすることができます。

threshold_queuesize しきい値キュー・サイズ : モニター・エレメント

キューのしきい値におけるキューのサイズ。このサイズを超えようとすると、しきい値違反が発生します。キューのしきい値以外では、この値は 0 です。

表 1216. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-

使用法

このエレメントを使用すると、しきい値の違反が発生した時点でのこのしきい値のキューにおけるアクティビティーまたは接続の数を判別できます。

thresholdid しきい値 ID : モニター・エレメント

しきい値違反レコードを適用するしきい値か、キュー統計の収集対象のしきい値を識別します。

表 1217. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-
統計	event_qstats	-

使用法

このモニター・エレメントを他のアクティビティー履歴モニター・エレメントと一緒に使用すると、しきい値キューの分析またはしきい値に違反したアクティビティーの分析をすることができます。

time_completed 完了時刻 : モニター・エレメント

このアクティビティー・レコードにより記述されているアクティビティーが実行を完了した時刻。このエレメントは、ローカル・タイム・スタンプです。

このフィールドは、メモリーの制約のために完全なアクティビティー・レコードを表イベント・モニターに書き込むことができなかった場合、またはアクティビティーが進行中にキャプチャーされた場合に、値が「0000-00-00-00.00.00.000000」になります。

エレメント ID
time_completed

エレメント・タイプ
情報

表 1218. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

使用法

このエレメントを他のアクティビティー履歴エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。

time_created 作成時刻 : モニター・エレメント

ユーザーが、このアクティビティー・レコードにより記述されているアクティビティーをサブミットした時刻。このエレメントは、ローカル・タイム・スタンプです。

表 1219. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

使用法

このエレメントを他のアクティビティー履歴エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。

time_of_violation 違反時刻 : モニター・エレメント

このしきい値違反レコードに記述されているしきい値違反が発生した時刻。このエレメントは、ローカル・タイム・スタンプです。

表 1220. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
しきい値違反	event_thresholdviolations	-

使用法

このエレメントを他のしきい値違反モニター・エレメントと一緒に使用すると、しきい値違反の分析をすることができます。

time_stamp スナップショット時刻

データベース・システム・モニター情報が収集された日時。

表 1221. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	collected	基本

使用法 このエレメントを使用すると、継続的な分析作業でその結果をファイルやデータベースに保管してある場合は、データを時系列的に関連付けることができます。

time_started 開始時刻 : モニター・エレメント

このアクティビティー・レコードにより記述されているアクティビティーが実行を開始した時刻。このエレメントは、ローカル・タイム・スタンプです。

表 1222. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-

使用法

このエレメントを他のアクティビティー履歴エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。

time_zone_disp 時間帯変位

グリニッジ標準時 (GMT) と現地時間帯の差を示す秒数。

表 1223. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	collected	基本

使用法 データベース・システム・モニターから報告される時刻はすべて GMT であり、現地時間はこの差に基づいて計算されます。

top ヒストグラム・ビンの最上位 : モニター・エレメント

ヒストグラム・ビンの範囲の包括的最上端。このモニター・エレメントの値は、次のヒストグラム・ビンの範囲の排他的最下端でもあります。

表 1224. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_histogrambin	-

使用法

このエレメントと対応する **bottom** エレメントを一緒に使用して、ヒストグラム中のビンの範囲を判別します。

tot_log_used_top 使用された最大合計ログ・スペース

使用された全ログ・スペースの最大量 (バイト単位)。

表 1225. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

表 1226. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 このエレメントは、ユーザーが割り振った 1 次ログ・スペースの量を評価するときに利用できます。このエレメントの値と割り振った 1 次ログ・スペースの量を比較すると、構成パラメーターの設定値を評価するときに利用できます。割り振った 1 次ログ・スペースは、次の公式で計算できます。

```
logprimary x logfilsiz x 4096 (下記の注を参照)
```

このエレメントと *sec_log_used_top* および *sec_logs_allocated* を組み合わせて使用すると、 2 次ログの依存度を示すことができます。

この値には、1 次と 2 次の両方のログ・ファイルに使用されるスペースが含まれます。

次の構成パラメーターの調整が必要になります。

- logfilsiz
- logprimary
- logsecond

注: データベース・システム・モニター情報はバイト単位で示されますが、構成パラメーターは各 4K バイトのページ単位で設定されます。

total_act_time - 合計アクティビティー時間 : モニター・エレメント

アクティビティーの実行にかかった時間の合計。この値はミリ秒単位で示されます。

表 1227. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

total_act_wait_time - 合計アクティビティー待機時間 : モニター・エレメント

アクティビティーの処理中に、DB2 データベース・サーバー内での待機にかかった合計時間。値はミリ秒単位で示されます。

表 1228. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	COLLECT ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

使用法

このエレメントを **total_act_request_time** エレメントと一緒に使用して、データ・サーバーがアクティビティーの処理に費やす時間のパーセンテージを判別します。

```
(total_act_request_time - total_act_wait_time)/(total_act_request_time) =
 % of time data server is actively working on activity
```

total_app_rqst_time - 合計アプリケーション要求時間 : モニター・エレメント

アプリケーション要求のためにかかった経過時間の合計。これは、アプリケーション要求を実行するサーバー上でコーディネーター・エージェントが費やした合計時間です。

表 1229. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

使用法

このモニター・エレメントを使用して、DB2 データ・サーバーでアプリケーション要求が費やした時間を判別します。この値は、観察されたパフォーマンス上の問題の原因がデータ・サーバーであるかどうかを判別するために利用できます。

例えば、アプリケーションで問題が生じ、回復するのに 20 分かかるとユーザーが報告する場合、合計アプリケーション要求時間が 1 分と判別され、その接続について現在進行中のアプリケーション要求がないのであれば、そのパフォーマンス上の問題は DB2 データ・サーバー以外のところにある可能性があります。

total_buffers_rcvd 受信された FCM バッファーの合計

node_number で識別されるノードが送信して、 GET SNAPSHOT コマンドを発行するノードで受信された FCM バッファーの合計数 (*db2nodes.cfg* ファイル参照)。

表 1230. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	fcm_node	基本

使用法 このエレメントを使用すると、現行ノードとリモート・ノードの間のトラフィック・レベルを測定できます。このノードから受信した FCM バッファーの数が多い場合は、データベースを再分散するか、または表を移動してノード間のトラフィックを少なくしてください。

total_buffers_sent 送信された FCM バッファーの合計

GET SNAPSHOT コマンドを発行するノードから *node_number* で識別されるノードに送信された FCM バッファーの合計数 (*db2nodes.cfg* ファイル参照)。

表 1231. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	fcm_node	基本

使用法 このエレメントを使用すると、現行ノードとリモート・ノードの間のトラフィック・レベルを測定できます。このノードに送信される FCM バッファーの数が多い場合は、データベースを再分散するか、または表を移動してノード間のトラフィックを少なくしてください。

total_cons データベース活動化以降の接続

最初の接続、活動化、または最後のリセット (コーディネーター・エージェント) 以降のデータベースへの接続数を示します。

表 1232. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
データベース	dbase_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1233. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

使用法 このエレメントと db_conn_time および db2start_time モニター・エレメントを組み合わせて使用すると、アプリケーションがデータベースに接続する頻度を計算できます。

接続の頻度が少ない場合は、他のアプリケーションに接続する前に、ACTIVATE DATABASE コマンドを使用して、データベースを活動化することもできます。これは、初めてデータベースに接続する際に時折生じる追加のオーバーヘッド (初期バッファー・プール割り振りなど) のためです。こうすると、それ以後の接続処理の速度を上げることができます。

注: このエレメントをリセットすると、値はゼロではなく、現在接続中のアプリケーションの数に設定されます。

total_cpu_time - 合計 CPU 時間 : モニター・エレメント

DB2 内で使用された CPU 時間の合計。ユーザーおよびシステム両方の CPU 時間の合計を表します。この値はマイクロ秒単位で示されます。

表 1234. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 1235. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE

表 1235. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	system_metrics 文書に報告されます。	-

total_exec_time ステートメント実行の経過時間

SQL キャッシュ内の特定のステートメントの実行に要した合計時間 (秒およびマイクロ秒単位)。

表 1236. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
動的 SQL	dynsql	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを num_executions とともに使用して、ステートメントの平均経過時間を判別し、 SQL の調整によって最も望ましい影響を受ける SQL ステートメントを識別します。このエレメントの内容を評価する際は、num_compilation も考慮する必要があります。

total_move_time 合計エクステント移動時間 : モニター・エレメント

表スペースのリバランス・プロセス中に移動したすべてのエクステントの移動時間の合計 (ミリ秒)。

表 1237. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_EXTENT_MOVEMENT_STATUS - エクステントの移動の進行状況メトリックの取得	常に収集される

total_hash_joins ハッシュ結合の合計

実行されたハッシュ結合の合計数。

エレメント ID
total_hash_joins

エレメント・タイプ
カウンター

表 1238. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1239. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 データベースまたはアプリケーション・レベルで、この値と hash_join_overflows および hash_join_small_overflows を組み合わせて使用すると、ソート・ヒープ・サイズを適度に変更することがハッシュ結合に有効かどうかを判別できます。

total_hash_loops ハッシュ・ループの合計

ハッシュ結合の単一パーティションが、使用可能なソート・ヒープ・スペースよりも大きかったときの合計回数。

表 1240. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1241. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 このエレメントの値は、ハッシュ結合が効率的に実行されていないことを示します。ソート・ヒープ・サイズが小さすぎるか、またはソート・ヒープしきい値が小さすぎることを示します。この値とその他のハッシュ結合変数を組み合わせて使用すると、ソート・ヒープ・サイズ (*sortheap*) とソート・ヒープしきい値 (*sheapthres*) の構成パラメーターを調整できます。

total_log_available 使用可能なログの合計

非コミット・トランザクションによって使用されていない、データベース内のアクティブ・ログ・スペースの量 (バイト単位)。

表 1242. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法

このエレメントを total_log_used とともに使用して、ログ・スペースを使い果たすことを避けるために以下の構成パラメーターを調整する必要があるかどうかを判別します。

- logfilsiz
- logprimary
- logsecond

total_log_available の値が 0 まで下がった場合、SQL0964N が返されます。上記の構成パラメーターの値を大きくするか、あるいは COMMIT、ROLLBACK または FORCE APPLICATION によって最も古いトランザクションを終了する必要があります。

logsecond が -1 に設定されていると、このエレメントには SQLM_LOGSPACE_INFINITE が含まれます。

注: データベース・システム・モニター情報はバイト単位で示されますが、構成パラメーターは各 4K バイトのページ単位で設定されます。

total_log_used 使用されているログ・スペースの合計

データベースで現在使用されているアクティブ・ログ・スペースの合計量 (バイト単位)。

表 1243. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 このエレメントを total_log_available とともに使用して、ログ・スペースを使い果たすことを避けるために以下の構成パラメーターを調整する必要があるかどうかを判別します。

- logfilsiz
- logprimary
- logsecond

注: データベース・システム・モニター情報はバイト単位で示されますが、構成パラメーターは各 4K バイトのページ単位で設定されます。

total_olap_funcs OLAP 関数の合計数 : モニター・エレメント

実行された OLAP 関数の合計数。

表 1244. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1245. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-

表 1245. イベント・モニター情報 (続き)

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	-

使用法

データベースまたはアプリケーション・レベルで、この値と olap_func_overflows を組み合わせて使用すると、ソート・ヒープ・サイズを適度に増やすことが、多くの割合の OLAP 関数に有効かどうかを判別できます。

total_rqst_mapped_in - マッピングの際の要求の合計 : モニター・エレメント

再マップしきい値または作業アクション・セットによって、このサービス・サブクラスにマップする際の要求の合計数。

表 1246. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1247. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE

total_rqst_mapped_out - マッピングの際に除外された要求の合計 : モニター・エレメント

再マップしきい値または作業アクション・セットによって、このサービス・サブクラスからマッピングの際に除外された要求の合計数。

表 1248. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1249. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE

total_rqst_time - 合計要求時間 : モニター・エレメント

要求の処理にかかった時間の合計。この値はミリ秒単位で報告されます。

表 1250. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

使用法

total_sec_cons 2 次接続

サブエージェントがノードのデータベースに接続した数。

エレメント ID

total_sec_cons

エレメント・タイプ

カウンター

表 1251. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本

使用法 このエレメントと total_cons、db_conn_time、および db2start_time のモニター・エレメントを組み合わせて使用すると、各アプリケーションがデータベースに接続した頻度を計算できます。

total_section_sorts - セクションのソートの合計: モニター・エレメント

セクションの実行 (クライアント・アプリケーションにより発行される SQL ステートメントによって生成される、コンパイルされた照会プランの実行) 中に実行されたソートの合計回数。

表 1252. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

使用法

このエレメントを total_section_sort_time と組み合わせて使用して、セクションの実行中にソートの実行にかかる平均時間を計算します。

アクティビティーおよびパッケージ・キャッシュ・レベルでこのエレメントを使用すると、多数のソート操作を実行するステートメントを識別できます。このようなステートメントは、さらに調整を行ってソート回数を少なくすると利点がありま

す。また、EXPLAIN ステートメントを使用すると、 1 つのステートメントが実行するソートの回数を識別できます。

total_section_sort_proc_time - セクションのソート処理時間の合計 : モニター・エレメント

セクションの実行 (クライアント・アプリケーションにより発行される SQL ステートメントによって生成される、コンパイルされた照会プランの実行) 中にソートを実行するのにかかった処理 (待機以外の) 時間の合計。

表 1253. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

使用法

システム・レベルでこのエレメントと total_section_sorts を組み合わせて使用すると、セクションの実行中の平均ソート処理時間 (待機を含まない) を計算できます。これは、ソートがパフォーマンス上の問題となっているかどうかを表します。

アクティビティー・レベルでこのエレメントを使用すると、ソート時間の多いステートメントを識別できます。このようなステートメントは、さらに調整を行ってソ

ート時間を少なくすると効率が上がります。

total_section_sort_time - セクションのソート時間の合計 : モニター・エレメント

セクションの実行 (クライアント・アプリケーションにより発行される SQL ステートメントによって生成される、コンパイルされた照会プランの実行) 中にソートを実行するのにかかった時間の合計。

表 1254. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

使用法

システム・レベルでこのエレメントと total_section_sorts を組み合わせて使用すると、セクションの実行中の平均ソート時間を計算できます。これは、ソートがステートメントのパフォーマンス上の問題となっているかどうかを表します。

注: total_section_sort_time エレメントには、待機時間と処理時間の両方が含まれます。(total_section_sort_time - total_section_sort_proc_time) が大きい場合、ソートは多くの時間を待機に費やしていることになります。例えば、ソートが頻繁にディスク

にスピルする場合、入出力待機のために total_section_sort_time が増加します。この時間は、ソートをアクティブに処理している時間のみをカウントする total_section_sort_proc_time には現れません。この場合、パフォーマンスを改善するためにソート・メモリーの調整を考慮できます。

アクティビティー・レベルでこのエレメントを使用すると、ソート時間の多いステートメントを識別できます。このようなステートメントは、さらに調整を行ってソート時間を少なくすると効率が上がります。

total_sort_time - ソート時間合計 : モニター・エレメント

実行されたすべてのソートの合計経過時間。この値はミリ秒単位で報告されます。

表 1255. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	Sort
アプリケーション	appl	Sort
アプリケーション	stmt	Sort
動的 SQL	dynsql	Sort

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1256. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-
ステートメント	event_stmt	-
アクティビティー	event_activity	ステートメント、ソート

使用法

データベースまたはアプリケーションのレベルでこのエレメントと **total_sorts** を組み合わせて使用すると、平均ソート時間を計算できます。平均ソート時間は、ソートがパフォーマンス上の問題となっているかどうかを表します。

ステートメント・レベルでこのエレメントを使用すると、ソート時間の多いステートメントを識別できます。このようなステートメントは、さらに調整を行ってソート時間を少なくすると効率が上がります。

このカウントには、関連操作のために作成される一時表のためのソート時間も含まれています。このカウントは、1 ステートメント、1 アプリケーション、または 1 つのデータベースにアクセスするすべてのアプリケーションについて情報を提供します。

経過時間を示すモニター・エレメントを使用するときは、次のことを考慮してください。

1. 経過時間は、システム負荷の影響を受けるので、実行する処理数が多くなると、この経過時間の値は大きくなる。

2. このモニター・エレメントをデータベース・レベルで計算する場合、データベース・システム・モニターはアプリケーション・レベルの時間を合計します。この場合、同時に複数のアプリケーション処理が実行されていることがあるので、データベース・レベルでは時間が二重に計算されます。

   データベース・レベルで意味のあるデータを得るためには、データを低いレベルに正規化する必要があります。以下に例を示します。

   ```
   total_sort_time / total_sorts
   ```

   これは、ソート当たりの平均経過時間に関する情報を示しています。

total_sorts - ソート合計 : モニター・エレメント

実行されたソートの合計数。

表 1257. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 1258. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1259. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
接続	event_conn	-
ステートメント	event_stmt	-
アクティビティー	event_activity	ステートメント、ソート

使用法

データベースまたはアプリケーションのレベルでは、この値と **sort_overflows** を組み合わせて使用すると、ヒープ・スペースをさらに必要とするソートのパーセンテージを計算できます。さらに、**total_sort_time** と組み合わせると、平均ソート時間を計算できます。

ソート合計数に対してソートのオーバーフロー回数が少ない場合は、ソート・ヒープ・サイズを大きくしても、バッファー・サイズを極端に大きくしなければ、パフォーマンスに影響を与えることはほとんどありません。

ステートメント・レベルでこのエレメントを使用すると、多数のソート操作を実行するステートメントを識別できます。このようなステートメントは、さらに調整を行ってソート回数を少なくすると利点があります。また、SQL EXPLAIN ステートメントを使用すると、 1 つのステートメントが実行するソートの回数を識別できます。

total_sys_cpu_time ステートメントのシステム CPU の合計時間 : モニター・エレメント

SQL ステートメントに使われたシステム CPU 時間の合計。

表 1260. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
動的 SQL	dynsql	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントをステートメント実行の経過時間およびステートメントのユーザー CPU の合計とともに使用して、最もコストがかかるステートメントを評価します。

total_sorts - ソート合計 : モニター・エレメント

実行されたソートの合計数。

表 1261. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

表 1262. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1263. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
データベース	event_db	-
接続	event_conn	-
ステートメント	event_stmt	-
アクティビティー	event_activity	ステートメント、ソート

使用法

データベースまたはアプリケーションのレベルでは、この値と **sort_overflows** を組み合わせて使用すると、ヒープ・スペースをさらに必要とするソートのパーセンテージを計算できます。さらに、**total_sort_time** と組み合わせると、平均ソート時間を計算できます。

ソート合計数に対してソートのオーバーフロー回数が少ない場合は、ソート・ヒープ・サイズを大きくしても、バッファー・サイズを極端に大きくしなければ、パフォーマンスに影響を与えることはほとんどありません。

ステートメント・レベルでこのエレメントを使用すると、多数のソート操作を実行するステートメントを識別できます。このようなステートメントは、さらに調整を行ってソート回数を少なくすると利点があります。また、SQL EXPLAIN ステートメントを使用すると、 1 つのステートメントが実行するソートの回数を識別できます。

total_usr_cpu_time ステートメントのユーザー CPU の合計時間 : モニター・エレメント

SQL ステートメントに使われたユーザー CPU 時間の合計。

表 1264. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
動的 SQL	dynsql	ステートメント

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法

このエレメントをステートメント実行の経過時間とともに使用して、最も実行時間が長いステートメントを評価します。

total_wait_time - 合計待機時間 : モニター・エレメント

DB2 データベース・サーバー内での待機にかかった合計時間。値はミリ秒単位で示されます。

表 1265. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

使用法

データベース・サーバーがアクティブに要求を処理するのに費やす時間のパーセンテージを判別するには、次の比率を使用します。

```
(total_rqst_time - total_wait_time) / total_rqst_time
```

client_idle_wait_time モニター・エレメントの値は、**total_wait_time** モニター・エレメントの値に含まれません。 **total_wait_time** エレメントはデータベース・サーバーが要求を処理している間の待機にかかった時間のみを表します。

tpmon_acc_str TP モニター・クライアント・アカウンティング・ストリング : モニター・エレメント

この接続で sqleseti API が発行された場合に、ロギングおよび診断の目的でターゲット・データベースに渡されたデータです。この接続、作業単位、またはアクティビティーの CLIENT_ACCTNG 特殊レジスターの現行値。

このモニター・エレメントは、**client_acctng** モニター・エレメントと同義です。 **client_acctng** モニター・エレメントは、DB2 バージョン 9.7 で導入された未フォー

マットの表に書き込む表関数およびイベント・モニターのモニターに使用されます。**tpmon_acc_str** モニター・エレメントは、表、ファイルおよびパイプに書き込むスナップショット・モニターおよびイベント・モニターに使用されます。

表 1266. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
DCS アプリケーション	dcs_appl	基本

表 1267. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-
デッドロック	event_dlconn	-
トランザクション	event_xact	-

使用法

このエレメントは、問題判別およびアカウンティングの目的で使用します。

tpmon_client_app TP モニター・クライアント・アプリケーション名 : モニター・エレメント

この接続で sqleseti API が発行された場合に、トランザクションを実行中のサーバー・トランザクション・プログラムを示します。この接続、作業単位、またはアクティビティーの CLIENT_APPLNAME 特殊レジスターの現行値。

このモニター・エレメントは、**client_applname** モニター・エレメントと同義です。**client_applname** モニター・エレメントは、DB2 バージョン 9.7 で導入された未フォーマットの表に書き込む表関数およびイベント・モニターのモニターに使用されます。**tpmon_client_app** モニター・エレメントは、表、ファイルおよびパイプに書き込むスナップショット・モニターおよびイベント・モニターに使用されます。

表 1268. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
DCS アプリケーション	dcs_appl	基本

表 1269. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-
デッドロック	event_dlconn	-
トランザクション	event_xact	-

使用法

このエレメントは、問題判別およびアカウンティングの目的で使用します。

tpmon_client_userid TP モニター・クライアント・ユーザー ID : モニター・エレメント

sqleseti API が使用された場合は、トランザクション・マネージャーが生成してサーバーに提供したクライアント・ユーザー ID。この接続、作業単位、またはアクティビティーの CLIENT_USERID 特殊レジスターの現行値。

このモニター・エレメントは、**client_userid** モニター・エレメントと同義です。**client_userid** モニター・エレメントは、DB2 バージョン 9.7 で導入された未フォーマットの表に書き込む表関数およびイベント・モニターのモニターに使用されます。**tpmon_client_userid** モニター・エレメントは、表、ファイルおよびパイプに書き込むスナップショット・モニターおよびイベント・モニターに使用されます。

表 1270. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本
DCS アプリケーション	dcs_appl	基本

表 1271. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-
デッドロック	event_dlconn	-
トランザクション	event_xact	-

使用法

アプリケーション・サーバーまたはトランザクション処理モニター環境でこのエレメントを使用すると、トランザクションの実行対象になっているエンド・ユーザーを識別できます。

tpmon_client_wkstn TP モニター・クライアント・ワークステーション名 : モニター・エレメント

この接続で sqleseti API が発行された場合に、クライアントのシステムまたはワークステーション (CICS EITERMID など) を示します。この接続、作業単位、またはアクティビティーの CLIENT_WRKSTNNAME 特殊レジスターの現行値。

このモニター・エレメントは、**client_wrkstnname** モニター・エレメントと同義です。 **client_wrkstnname** モニター・エレメントは、DB2 バージョン 9.7 で導入された未フォーマットの表に書き込む表関数およびイベント・モニターのモニターに使用されます。**tpmon_client_wkstn** モニター・エレメントは、表、ファイルおよびパイプに書き込むスナップショット・モニターおよびイベント・モニターに使用されます。

表 1272. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本

表 1272. スナップショット・モニター情報 (続き)

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl	基本

表 1273. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity	-
デッドロック	event_dlconn	-
トランザクション	event_xact	-

使用法

このエレメントを使用すると、ノード ID、端末 ID、またはその他の同様の ID を使用して、ユーザーのマシンを識別できます。

tq_cur_send_spills オーバーフローした表キュー・バッファーの現在数 : モニター・エレメント

一時表内にある表キュー・バッファーの現在の数。

表 1274. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	サブセクション	ステートメント

使用法 表キューへの書き込みを行っているエージェントは、複数のリーダーに行を送信している可能性があります。書き込みエージェントが行を送信したときに、相手のエージェントが行を受け付けず、別のエージェントが処理のために行を必要とすると、書き込みエージェントはバッファーを一時表にオーバーフローします。一時表にオーバーフローすると、ライターとほかのリーダーがともに処理を続行できるようになります。

オーバーフローした行は、読み取りエージェントが行を受け付ける準備ができると読み取りエージェントに送信されます。

この数値が大きく、照会が sqlcode -968 で失敗し、さらに db2diad.log 内のメッセージにより TEMP 表スペースの一時スペースがなくなったことを示す場合は、表キューのオーバーフローが原因の可能性があります。この場合、ほかのノードで問題が発生していることを示します (ロッキングなど)。この照会のすべてのパーティション上でスナップショットを取って、調査してください。

データをパーティション化する方法が原因で、照会によって多数のバッファーをオーバーフローすることが必要になる場合もあります。このような原因がある場合は、 TEMPORARY 表スペースにさらにディスクを追加する必要があります。

tq_id_waiting_on ノード上の表キュー待機

待機中のエージェント。

エレメント ID

tq_id_waiting_on

エレメント・タイプ

情報

表 1275. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	サブセクション	ステートメント

使用法 これはトラブルシューティングに使用できます。

tq_max_send_spills 表キュー・バッファー・オーバーフローの最大数

一時表にオーバーフローした表キュー・バッファーの最大数。

エレメント ID

tq_max_send_spills

エレメント・タイプ

水準点

表 1276. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	サブセクション	ステートメント

表 1277. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_subsection	-

使用法 一時表に書き込まれた表キュー・バッファーの最大数を示します。

tq_node_waited_for 表キュー上のノード待機

サブセクション状況の ss_status が「*受信待ち*」または「*送信待ち*」で、tq_wait_for_any が FALSE の場合は、エージェントが待機中のノード番号を示します。

エレメント ID

tq_node_waited_for

エレメント・タイプ

情報

表 1278. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	サブセクション	ステートメント

使用法 これはトラブルシューティングに使用できます。サブセクションが待機して

いるノード上でアプリケーションのスナップショットが必要になる場合があります。例えば、アプリケーションがそのノード上でロック待機になっている場合です。

tq_rows_read 表キューから読み取られた行数

表キューから読み取られた合計行数。

エレメント ID

tq_rows_read

エレメント・タイプ

カウンター

表 1279. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	サブセクション	ステートメント

表 1280. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_subsection	-

使用法 この数が増加していることをモニターが示していない場合は、処理は進行していません。

ノードごとにこの数値が大きく異なる場合は、使用率が過剰なノードと過少なノードがあります。

この数値が大きい場合は、ノード間で転送されるデータ量が多く、最適化によりアクセス・プランを改善できることを示します。

tq_rows_written 表キューに書き込まれた行数

表キューに書き込まれた合計行数。

エレメント ID

tq_rows_written

エレメント・タイプ

カウンター

表 1281. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	サブセクション	ステートメント

表 1282. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ステートメント	event_subsection	-

使用法 この数が増加していることをモニターが示していない場合は、処理は進行していません。

ノードごとにこの数値が大きく異なる場合は、使用率が過剰なノードと過少なノードがあります。

この数値が大きい場合は、ノード間で転送されるデータ量が多く、最適化によりアクセス・プランを改善できることを示します。

tq_tot_send_spills - オーバーフローした表キュー・バッファーの合計数 : モニター・エレメント

一時表にオーバーフローした表キュー・バッファーの合計数。

表 1283. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE

表 1284. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	subsection	ステートメント

表 1285. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
ステートメント	event_subsection	-

使用法

一時表に書き込まれた表キュー・バッファーの合計数を示します。詳しくは、**tq_cur_send_spills** モニター・エレメントを参照してください。

tq_wait_for_any 表キュー上のノード送信待機

このフラグは、サブセクションがノードからの行の受信を待っているためにサブセクションがブロックされていることを示すのに使用されます。

エレメント ID

tq_wait_for_any

エレメント・タイプ

情報

表 1286. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	サブセクション	ステートメント

使用法 ss_status が「*表キュー上でデータの受信待ち*」を示し、このフラグが TRUE の場合は、サブセクションはノードからの行の受信を待機中です。この場合は通常、 SQL ステートメントが待機中のエージェントにデータを渡すところまでまだ処理が進んでいないことを示します。例えば、書き込みエージェントがソートを実行中で、ソートが終了するまでは行を書き込まない場合があります。 db2expln の出力を使用して、エージェントが行の受信を待機している表キューに関連付けられたサブセクション番号を判別します。次に、サブセクションを実行している各ノードでスナップショットをとると、サブセクションの状況を確認できます。

ts_name - ロールフォワードされている表スペース : モニター・エレメント

現在ロールフォワードされている表スペースの名前。

エレメント ID

ts_name

エレメント・タイプ

情報

表 1287. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表スペース	rollforward	基本

使用法 ロールフォワードが進行中の場合は、このエレメントは関係する表スペースを示します。

uid_sql_stmts 実行された更新/挿入/削除 SQL ステートメント

実行された SQL UPDATE、INSERT、および DELETE ステートメントの数。

表 1288. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1289. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 このエレメントを使用すると、アプリケーション・レベルまたはデータベース・レベルのデータベース・アクティビティーのレベルを判別できます。

次の公式を使用すると、すべてのステートメントに対する UPDATE、INSERT および DELETE ステートメントの比率を計算できます。

```
  uid_sql_stmts
/ (static_sql_stmts + dynamic_sql_stmts )
```

この情報は、アプリケーションのアクティビティーおよびスループットの分析に役立ちます。

unread_prefetch_pages - 読み取り不能プリフェッチ・ページ : モニター・エレメント

プリフェッチャー読み取りが使用されなかったページ数を示します。

表 1290. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE

表 1291. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	バッファー・プール
表スペース	tablespace	バッファー・プール
バッファー・プール	bufferpool	バッファー・プール
アプリケーション	appl	バッファー・プール

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1292. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
表スペース	event_tablespace	-
接続	event_conn	-

使用法

ページ数が多い場合、プリフェッチャーは、使用されないページをバッファー・プール内に読み込むことで不必要な入出力を行うことがあります。

uow_comp_status 作業単位完了状況

作業単位の状況およびそれが停止したときの状況。

エレメント ID

uow_comp_status

エレメント・タイプ

情報

表 1293. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	作業単位
DCS アプリケーション	dcs_appl	基本

表 1294. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
トランザクション	event_xact	-

使用法 このエレメントを使用すると、作業単位が終了した原因がデッドロックによるものか、または異常終了によるものかを判別できます。次の原因が考えられます。

- コミット・ステートメントによりコミットされた。
- ロールバック・ステートメントによりロールバックされた。
- デッドロックによりロールバックされた。
- 異常終了によりロールバックされた。
- アプリケーションの正常終了によりコミットされた。
- 進行中であった作業単位に対する FLUSH EVENT MONITOR コマンドの結果が不明。

注: API ユーザーは、データベース・システム・モニターの定数の定義が含まれているヘッダー・ファイル (*sqlmon.h*) を参照してください。

uow_elapsed_time 最新の作業単位の経過時間

最後に完了した作業単位の実行経過時間。

エレメント ID

uow_elapsed_time

エレメント・タイプ

時間

表 1295. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	作業単位、タイム・スタンプ
DCS アプリケーション	dcs_appl	作業単位、タイム・スタンプ

使用法 作業単位の完了にかかる時間の標識として、このエレメントを使用します。

uow_id 作業単位 ID : モニター・エレメント

作業単位の ID。作業単位 ID は、アプリケーション・ハンドル内で固有です。

表 1296. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	常に収集される
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	常に収集される
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得	常に収集される

表 1297. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
作業単位	-	-
アクティビティー	event_activity	-
アクティビティー	event_activitystmt	-
アクティビティー	event_activityvals	-
しきい値違反	event_thresholdviolations	-

使用法

このエレメントを他のアクティビティー履歴エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。

さらにこのエレメントを **activity_id** および **appl_id** モニター・エレメントと一緒に使用すると、アクティビティーを一意的に識別できます。

uow_lock_wait_time - ロック待機中の作業単位の合計時間 : モニター・エレメント

この作業単位がロックの待機に要した合計経過時間。値はミリ秒単位で示されます。

エレメント ID
uow_lock_wait_time

エレメント・タイプ
カウンター

表 1298. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	作業単位

使用法 このエレメントは、リソース競合問題の重大度を判別するときに利用できます。

uow_log_space_used 使用されている作業単位ログ・スペース

モニター対象のアプリケーションで現行作業単位に使用されているログ・スペースの量 (バイト単位)。

エレメント ID

uow_log_space_used

エレメント・タイプ

ゲージ

表 1299. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	作業単位

表 1300. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
トランザクション	event_xact	-

使用法 このエレメントを使用すると、作業単位レベルでのロギングの所要量を把握することができます。

uow_start_time 作業単位開始タイム・スタンプ

作業単位が最初にデータベース・リソースを要求した日時。

表 1301. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	作業単位、タイム・スタンプ
DCS アプリケーション	dcs_appl	作業単位、タイム・スタンプ

表 1302. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	-

使用法

このリソース要求は、その作業単位で SQL ステートメントを初めて実行したときに発生します。

- 最初の作業単位の場合は、**conn_complete_time** の後の最初のデータベース要求 (SQL ステートメントの実行) の時刻。

- その後の作業単位の場合は、前回の COMMIT または ROLLBACK の後の最初のデータベース要求 (SQL ステートメントの実行) の時刻。

注: 「*SQL リファレンス*」は、作業単位の境界を COMMIT または ROLLBACK のポイントとして定義します。

データベース・システム・モニターでは、 COMMIT/ROLLBACK とその作業単位定義から出される次の SQL ステートメントまでの経過時間を除外します。この測定方式により、データベース・マネージャーがデータベース要求の処理に要する時間を、その作業単位の最初の SQL ステートメント以前にアプリケーション・ロジック内で要する時間とは切り離して反映します。作業単位の経過時間には、作業単位内で SQL ステートメント間のアプリケーション・ロジックを実行する時間が含まれます。

このエレメントと **uow_stop_time** モニター・エレメントを組み合わせると、作業単位の合計経過時間を計算できます。**prev_uow_stop_time** モニター・エレメントと組み合わせると、作業単位間にアプリケーションで要した時間を計算できます。

uow_stop_time と **prev_uow_stop_time** モニター・エレメントを組み合わせると、*SQL リファレンス*の定義による作業単位の経過時間を計算できます。

uow_status 作業単位の状況

作業単位の状況。

エレメント ID

uow_status

エレメント・タイプ

情報

表 1303. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
トランザクション	event_xact	-

使用法 このエレメントを使用すると、作業単位の状況を判別できます。 API ユーザーは、データベース・システム・モニターの定数の定義が含まれているヘッダー・ファイル sqlmon.h を参照してください。

uow_stop_time 作業単位停止タイム・スタンプ : モニター・エレメント

最新の作業単位が完了した日時。これが起こるのはデータベースの変更がコミットまたはロールバックされたときです。

表 1304. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl	作業単位、タイム・スタンプ
DCS アプリケーション	dcs_appl	作業単位、タイム・スタンプ

表 1305. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	-

使用法

このエレメントと **prev_uow_stop_time** モニター・エレメントを組み合わせて使用すると、COMMIT/ROLLBACK ポイント間の合計経過時間を計算できます。**uow_start_time** モニター・エレメントと組み合わせると、前回の作業単位の経過時間を計算できます。

タイム・スタンプの内容は、次のように設定されます。

- アプリケーションが 1 つの作業単位を完了し、(**uow_start_time** モニター・エレメントで定義されたように) 新しい作業単位をまだ開始していない場合、このエレメントは有効な非ゼロタイム・スタンプをレポートします。
- アプリケーションが作業単位を実行中の場合は、このエレメントがゼロをレポートします。
- アプリケーションがデータベースに初めて接続すると、このエレメントは **conn_complete_time** モニター・エレメントの値に設定されます。

新しい作業単位が開始すると、このエレメントの内容は、**prev_uow_stop_time** モニター・エレメントに移動します。

uow_total_time_top - UOW 合計時間の最上位 : モニター・エレメント

将来の利用のために予約されています。

表 1306. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats	-
統計	event_scstats	-

使用法

将来の利用のために予約されています。

update_sql_stmts 更新回数

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時点か、またはデータベース・モニター・カウンターの最後のリセット時点以降に、いずれかのアプリケーションに代わってフェデレーテッド・サーバーがこのデータ・ソースに UPDATE ステートメントを発行した合計回数が含まれています。

表 1307. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	基本
アプリケーション	appl_remote	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

使用法 このエレメントを使用すると、フェデレーテッド・サーバーまたはアプリケーションによりこのデータ・ソースに対して行われたデータベース・アクティビティーのレベルを判別できます。

このエレメントを使用すると、次の公式を使用して、フェデレーテッド・サーバーまたはアプリケーションによるこのデータ・ソースへの書き込みアクティビティーのパーセンテージも判別できます。

```
書き込みアクティビティー =
  (INSERT ステートメント + UPDATE ステートメント +
  DELETE ステートメント) /
  (SELECT ステートメント + INSERT ステートメント +
  UPDATE ステートメント +
  DELETE ステートメント)
```

update_time 更新応答時間

このエレメントには、フェデレーテッド・サーバー・インスタンスの開始時点か、またはデータベース・モニター・カウンターの最後のリセット時点以降に、このフェデレーテッド・サーバー・インスタンス上で実行されているすべてのアプリケーションまたは単一アプリケーションからの UPDATE に対して、このデータ・ソースが応答に要した合計時間が含まれています (ミリ秒単位)。

表 1308. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase_remote	タイム・スタンプ
アプリケーション	appl_remote	タイム・スタンプ

スナップショット・モニターの場合、このカウンターはリセットできます。

応答時間とは、フェデレーテッド・サーバーが UPDATE ステートメントをデータ・ソースにサブミットしてからデータ・ソースが UPDATE を処理したことをフェデレーテッド・サーバーに応答するまでの時間です。

使用法 このエレメントを使用すると、このデータ・ソースに対する UPDATE が処理されるのを待機するために生じた実際の時間を判別できます。この情報は、キャパシティー・プランニングおよびチューニングを行うときに便利です。

user_cpu_time ユーザー CPU 時間

データベース・マネージャーのエージェント・プロセス、作業単位、またはステートメントで使用されたユーザー CPU 時間の合計 (秒およびマイクロ秒単位)。

ステートメント・モニター・スイッチまたはタイム・スタンプ・スイッチがオンになっていない場合は、このエレメントは収集されず、代わりに -1 が書き込まれます。

表 1309. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
接続	event_conn	-
トランザクション	event_xact	-
ステートメント	event_stmt	-
アクティビティー	event_activity	-

使用法 このエレメントと CPU 時間に関連する他のエレメントを組み合わせて使用すると、アプリケーション内のアクティビティーのレベルがわかります。また、さらに調整するとその効果が得られる可能性があるアプリケーションを識別できます。

注: ユーザーのオペレーティング・システムでこの情報が利用できないとき、このエレメントは 0 に設定されます。

utility_dbname ユーティリティーで操作されるデータベース

ユーティリティーで操作されているデータベース。

表 1310. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	utility_info	基本

utility_description ユーティリティー記述

ユーティリティーが実行している作業を簡潔に示す記述。例えば、rebalance 呼び出しに「Tablespace ID: 2」が含まれている場合、このリバランサーが ID 2 の表スペースに対して機能していることを示します。このフィールドのフォーマットはユーティリティー・クラスに依存し、リリースごとに変更される可能性があります。

表 1311. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	utility_info	基本

utility_id ユーティリティー ID

ユーティリティー呼び出しに対応するユニーク ID。

表 1312. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

表 1313. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	utility_info	基本

utility_invoker_type - ユーティリティー呼び出し側タイプ

このエレメントは、ユーティリティーが起動された方法を説明しています。

表 1314. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	utility_info	基本

使用法 このエレメントを使用すると、ユーティリティーが呼び出された方法を判別できます。例えば、このエレメントを使用すると、ユーティリティーが DB2 によって自動的に呼び出されたか、またはユーザーによって呼び出されたかを判別できます。以下のリストにある、このエレメントの値は、sqlmon.h で定義されています。

API 定数	ユーティリティー
SQLM_UTILITY_INVOKER_USER	ユーティリティーはユーザーによって呼び出されました。
SQLM_UTILITY_INVOKER_AUTO	ユーティリティーは DB2 によって自動的に呼び出されました。

utility_priority ユーティリティー優先度

ユーティリティー優先度は、スロットルされたピアに関連したスロットル・ユーティリティーの相対的な重要度を指定します。優先度 0 は、ユーティリティーがスロットルされずに実行されることを意味します。非ゼロの優先度は 1 から 100 の範囲にする必要があります。 100 は最高の優先度、1 は最低の優先度です。

表 1315. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	utility_info	基本

utility_start_time ユーティリティー開始時刻

現在のユーティリティーがもともと呼び出された日付と時刻。

表 1316. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	utility_info	基本

utility_state - ユーティリティー状態

このエレメントは、ユーティリティーの状態を示します。

エレメント ID

utility_state

エレメント・タイプ

情報

表 1317. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	utility_info	基本

使用法 このエレメントを使用して、アクティブ・ユーティリティーの状態を判別できます。以下のリストにある、このフィールドの値は、sqlmon.h で定義されています。

API 定数	説明
SQLM_UTILITY_STATE_EXECUTE	ユーティリティーは実行されています。
SQLM_UTILITY_STATE_WAIT	ユーティリティーは、進行を再開する前にイベントが発生するのを待機しています。
SQLM_UTILITY_STATE_ERROR	ユーティリティーは、エラーを検出しました。

valid セクション妥当性インディケーター : モニター・エレメント

動的 SQL ステートメント・セクションが有効かどうかを示します。 静的 SQL ステートメントの場合、このモニター・エレメントの値は常に Y です。

表 1318. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュの SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

使用法

このモニター・エレメントの有効値は Y および N です。システムが次回使用される時に、無効なセクションはシステムによって暗黙に準備されます。

utility_type ユーティリティー・タイプ

ユーティリティーのクラス。

表 1319. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース・マネージャー	utility_info	基本

使用法

このエレメントの値は、sqlmon.h で定義された ″SQLM_UTILITY_″ の名前で始まる定数になります。

vectored_ios - ベクトル化入出力要求数 : モニター・エレメント

ベクトル化入出力要求の数です。より厳密には、 DB2 がバッファー・プールのページ域に対してページの順次プリフェッチを実行する回数です。

表 1320. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_BUFFERPOOL 表関数 - バッファー・プール・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_TABLESPACE 表関数 - 表スペース・メトリックの取得	DATA OBJECT METRICS BASE
MON_GET_CONTAINER 表関数 - 表スペース・コンテナー・メトリックの取得	DATA OBJECT METRICS BASE

表 1321. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
バッファー・プール	bufferpool	バッファー・プール

使用法

このエレメントを使用すると、ベクトル化入出力の実行頻度を判別できます。ベクトル化入出力要求の数がモニターされるのは、順次プリフェッチの実行中だけです。

version モニター・データのバージョン

イベント・モニター・データ・ストリームを作成したデータベース・マネージャーのバージョン。

表 1322. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
イベント・ログ・ヘッダー	event_log_header	-

使用法

イベント・モニターが使用するデータ構造は、データベース・マネージャーのリリースによって異なっている場合があります。そのため、モニター・アプリケーションは、受信するデータを処理できるかどうかを判別するために、データ・ストリームのバージョンを確認する必要があります。

このリリースでは、このエレメントは API 定数の SQLM_DBMON_VERSION9_5 に設定されています。

wlm_queue_assignments_total - ワークロード・マネージャー合計キュー割り当て : モニター・エレメント

アクティビティーが WLM しきい値によってキューに入れられた回数。

表 1323. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

wlm_queue_time_total - ワークロード・マネージャー合計キュー時間 : モニター・エレメント

WLM キューイングしきい値の待機にかかった時間。この値はミリ秒単位で示されます。

表 1324. 表関数モニター情報

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION 表関数 - 接続メトリックの取得	REQUEST METRICS BASE

表 1324. 表関数モニター情報 (続き)

表関数	モニター・エレメント収集レベル
MON_GET_CONNECTION_DETAILS 表関数 - 接続メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS 表関数 - サービス・サブクラス・メトリックの取得	REQUEST METRICS BASE
MON_GET_SERVICE_SUBCLASS_DETAILS 表関数 - サービス・サブクラス・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	REQUEST METRICS BASE
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	REQUEST METRICS BASE
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得 (DETAILS XML 文書に報告されます)	REQUEST METRICS BASE
MON_GET_ACTIVITY_DETAILS 表関数 - 完全なアクティビティー詳細の取得 (DETAILS XML 文書に報告されます)	ACTIVITY METRICS BASE
MON_GET_PKG_CACHE_STMT 表関数 - パッケージ・キャッシュ内の SQL ステートメント・アクティビティー・メトリックの取得	ACTIVITY METRICS BASE

wlo_completed_total 完了したワークロード・オカレンスの合計 : モニター・エレメント

最後にリセットしてから完了するワークロード・オカレンスの数。

表 1325. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wlstats	-

使用法

このエレメントを使用すると、処理をシステムに移動させている特定のワークロードのオカレンスの数を判別できます。

work_action_set_id 作業アクション・セット ID : モニター・エレメント

この統計レコードが適用される作業アクション・セットの ID。

表 1326. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_histogrambin	-
統計	event_wcstats	-

使用法

このエレメントを他のアクティビティー履歴エレメントと一緒に使用すると、アクティビティーの動作の分析をすることができます。あるいは、他の統計エレメントと一緒に使用すると、作業クラスの動作の分析をすることができます。

work_action_set_name 作業アクション・セット名 : モニター・エレメント

このイベントの一部として示された統計が関連付けられた作業アクション・セットの名前。

表 1327. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	-
統計	event_wcstats	-

使用法

このエレメントと **work_class_name** エレメントを組み合わせて使用すると、統計がこのレコードに示されている作業クラスを一意的に識別したり、統計がこのレコードに示されているしきい値キューのドメインである作業クラスを一意的に識別できます。

work_class_id 作業クラス ID : モニター・エレメント

この統計レコードが適用される作業クラスの ID。

表 1328. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_wcstats	-
統計	event_histogrambin	-

使用法

このエレメントを他の統計エレメントと一緒に使用すると、作業クラスの分析をすることができます。

work_class_name 作業クラス名 : モニター・エレメント

このイベントの一部として示された統計が関連付けられた作業クラスの名前。

表 1329. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
統計	event_qstats	-
統計	event_wcstats	-

使用法

このエレメントと **work_action_set_name** エレメントを組み合わせて使用すると、統計がこのレコードに示されている作業クラスを一意的に識別したり、統計がこのレコードに示されているしきい値キューのドメインである作業クラスを一意的に識別できます。

workload_id ワークロード ID : モニター・エレメント

ワークロードを一意的に識別する整数。

表 1330. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	常に収集される
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得	常に収集される

表 1331. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
アプリケーション	appl_info	基本

表 1332. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
作業単位	-	-
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
統計	event_wlstats	-
統計	event_histogrambin	-
アクティビティー	event_activity	-

使用法

この ID を使用して、このアクティビティー、アプリケーション、ヒストグラム・ビン、またはワークロード統計レコードが所属するワークロードを一意的に識別します。

workload_name ワークロード名 : モニター・エレメント

ワークロードの名前。

表 1333. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	常に収集される
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	常に収集される
MON_GET_WORKLOAD 表関数 - ワークロード・メトリックの取得	常に収集される
MON_GET_WORKLOAD_DETAILS 表関数 - ワークロード・メトリック詳細の取得	常に収集される

表 1334. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
ロッキング	-	-
作業単位	-	-
アクティビティー	event_activity (details_xml 文書に報告されます)	ACTIVITY METRICS BASE
統計	event_scstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
統計	event_wlstats (details_xml 文書に報告されます)	REQUEST METRICS BASE
作業単位	system_metrics 文書に報告されます。	-
統計	event_wlstats	-

使用法

統計イベント・モニターおよびワークロードの表関数において、ワークロード名は統計またはメトリックが収集されたり報告されるワークロードを識別します。 作業単位イベント・モニターおよび作業単位表関数の場合、ワークロード名は作業単位が関連付けられたワークロードを識別します。

ワークロード名を使用して、特定のワークロードに関する作業単位や情報のセットを識別します。

workload_occurrence_id ワークロード・オカレンス ID : モニター・エレメント

このアクティビティーが所属するワークロード・オカレンスの ID。

表 1335. 表関数モニター情報

表関数	モニター・エレメントの収集レベル
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	常に収集される
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得	常に収集される

表 1336. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
作業単位	-	-
アクティビティー	event_activity	-

使用法

これを使用すると、アクティビティーをサブミットしたワークロード・オカレンスを識別できます。

workload_occurrence_state - ワークロード・オカレンスの状態 : モニター・エレメント

ワークロード・オカレンスの状態。

表 1337. 表関数モニター情報

表関数	モニター・エレメントの収集コマンドおよびレベル
MON_GET_UNIT_OF_WORK 表関数 - 作業単位メトリックの取得	常に収集される
MON_GET_UNIT_OF_WORK_DETAILS 表関数 - 作業単位メトリック詳細の取得 (DETAILS XML 文書に報告されます)	常に収集される

使用法

可能な値は以下のとおりです。

DECOUPLED
ワークロード・オカレンスにコーディネーター・エージェントが割り当てられていません (コンセントレーターのケース)。

DISCONNECTPEND
ワークロード・オカレンスはデータベースから切断中です。

FORCED
ワークロード・オカレンスは強制されました。

INTERRUPTED
ワークロード・オカレンスは中断されました。

QUEUED
ワークロード・オカレンスのコーディネーター・エージェントは、Query Patroller またはワークロード管理のキューイングしきい値によってキューに入れられています。データベース・パーティション・フィーチャー (DPF) 環境では、この状態はコーディネーター・エージェントがしきい値チケットを取得するためにカタログ・パーティションへの RPC を行ったものの、まだ応答を受け取っていないことを示している可能性があります。

TRANSIENT
ワークロード・オカレンスは、サービス・スーパークラスにまだマップされていません。

UOWEXEC
ワークロード・オカレンスは要求の処理中です。

UOWWAIT
ワークロード・オカレンスはクライアントからの要求を待機しています。

x_lock_escals 排他ロック・エスカレーション数

ロックが複数の行ロックから 1 つの排他表ロックにエスカレートされた回数。または行に対する排他ロックにより表ロックが排他ロックになった回数。

エレメント ID
x_lock_escals

エレメント・タイプ
カウンター

表 1338. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1339. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-
トランザクション	event_xact	-

使用法 他のアプリケーションは排他ロックによって保留されているデータにアクセスすることができません。そのため、排他ロックはデータの並行性に影響を与える可能性があるため、それを追跡することは重要です。

アプリケーションが保留するロックの合計数がそのアプリケーションで使用可能なロック・リスト・スペースの最大量に達すると、ロックはエスカレー

トされます。使用可能なロック・リスト・スペースの量は、 *locklist* および *maxlocks* 構成パラメーターによって決まります。

1 つのアプリケーションが使用可能な最大ロック数に達して、エスカレートするロックがほかにない場合は、ほかのアプリケーションに割り振られているロック・リストのスペースが使用されます。ロック・リスト全体が満杯になるとエラーが起こります。

過度の排他ロック・エスカレーションが起こる場合の考えられる原因と対策については、『*lock_escals*』を参照してください。

共有ロックが十分にあるのに、アプリケーションは排他ロックを使用することがあります。共有ロックによってロック・エスカレーションの合計数を減らすことはできませんが、排他ロックのエスカレーションよりも共有ロックのエスカレーションのほうが望ましいと考えられます。

xda_object_pages XDA オブジェクト・ページ数

XML ストレージ・オブジェクト (XDA) データが消費するディスク・ページの数。

エレメント ID
xda_object_pages

エレメント・タイプ
情報

表 1340. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
表	table	基本

表 1341. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
表	event_table	-

使用法 このエレメントを使用すると、特定の表中の XML ストレージ・オブジェクト (XDA) データが消費する実際のスペースの量を表示できます。このエレメントと表イベント・モニターを組み合わせて使用すると、時間とともに XML ストレージ・オブジェクト・データが大きくなる比率を追跡できます。

xid トランザクション ID

トランザクション・マネージャーが 2 フェーズ・コミットのトランザクションで生成した、 (すべてのデータベースにおいて) ユニークなトランザクション ID。

表 1342. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
DCS アプリケーション	dcs_appl	作業単位

使用法 この ID を使用すると、トランザクション・マネージャーが生成したトランザクションと複数のデータベースに対して実行されたトランザクションを関

連付けることができます。トランザクション・マネージャーに問題があったときに、 2 フェーズ・コミット・プロトコルが関与するデータベース・トランザクションとトランザクション・マネージャーが発信したトランザクションを対応させて、問題の診断に利用できます。

xquery_stmts - 試行された XQuery ステートメント

アプリケーションまたはデータベースに対して実行される XQuery ステートメントの数。

エレメント ID

xquery_stmts

エレメント・タイプ

カウンター

表 1343. スナップショット・モニター情報

スナップショット・レベル	論理データ・グループ	モニター・スイッチ
データベース	dbase	基本
アプリケーション	appl	基本

スナップショット・モニターの場合、このカウンターはリセットできます。

表 1344. イベント・モニター情報

イベント・タイプ	論理データ・グループ	モニター・スイッチ
データベース	event_db	-
接続	event_conn	-

使用法 このエレメントを使用すると、固有の XQuery 言語要求のアクティビティーを測定することができます。これには xmlquery、xmltable、または xmlexist などの組み込み XQuery 言語要求は含まれません。

第 3 部 付録

付録 A. DB2 技術情報の概説

DB2 技術情報は、以下のツールと方法を介して利用できます。

- DB2 インフォメーション・センター
  - トピック (タスク、概念、およびリファレンス・トピック)
  - DB2 ツールのヘルプ
  - サンプル・プログラム
  - チュートリアル
- DB2 資料
  - PDF ファイル (ダウンロード可能)
  - PDF ファイル (DB2 PDF DVD に含まれる)
  - 印刷資料
- コマンド行ヘルプ
  - コマンド・ヘルプ
  - メッセージ・ヘルプ

注: DB2 インフォメーション・センターのトピックは、PDF やハードコピー資料よりも頻繁に更新されます。最新の情報を入手するには、資料の更新が発行されたときにそれをインストールするか、ibm.com にある DB2 インフォメーション・センターを参照してください。

技術資料、ホワイト・ペーパー、IBM Redbooks® 資料などのその他の DB2 技術情報には、オンライン (ibm.com) でアクセスできます。 DB2 Information Management ソフトウェア・ライブラリー・サイト (http://www.ibm.com/software/data/sw-library/) にアクセスしてください。

資料についてのフィードバック

DB2 の資料についてのお客様からの貴重なご意見をお待ちしています。DB2 の資料を改善するための提案については、db2docs@ca.ibm.com まで E メールを送信してください。DB2 の資料チームは、お客様からのフィードバックすべてに目を通しますが、直接お客様に返答することはありません。お客様が関心をお持ちの内容について、可能な限り具体的な例を提供してください。特定のトピックまたはヘルプ・ファイルについてのフィードバックを提供する場合は、そのトピック・タイトルおよび URL を含めてください。

DB2 お客様サポートに連絡する場合には、この E メール・アドレスを使用しないでください。資料を参照しても、DB2 の技術的な問題が解決しない場合は、お近くの IBM サービス・センターにお問い合わせください。

DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)

以下の表は、DB2 ライブラリーについて説明しています。DB2 ライブラリーに関する詳細な説明については、www.ibm.com/shop/publications/order にある IBM Publications Center にアクセスしてください。英語および翻訳された DB2 バージョン 9.7 のマニュアル (PDF 形式) は、 www.ibm.com/support/docview.wss?rs=71&uid=swg2700947 からダウンロードできます。

この表には印刷資料が入手可能かどうかが示されていますが、国または地域によっては入手できない場合があります。

資料番号は、資料が更新される度に大きくなります。資料を参照する際は、以下にリストされている最新版であることを確認してください。

注: *DB2 インフォメーション・センター*は、PDF やハードコピー資料よりも頻繁に更新されます。

表 1345. DB2 の技術情報

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
管理 API リファレンス	SC88-5883-00	入手可能	2009 年 8 月
管理ルーチンおよびビュー	SC88-5880-00	入手不可	2009 年 8 月
コール・レベル・インターフェース ガイドおよびリファレンス 第 1 巻	SC88-5885-00	入手可能	2009 年 8 月
コール・レベル・インターフェース ガイドおよびリファレンス 第 2 巻	SC88-5886-00	入手可能	2009 年 8 月
コマンド・リファレンス	SC88-5884-00	入手可能	2009 年 8 月
データ移動ユーティリティー ガイドおよびリファレンス	SC88-5903-00	入手可能	2009 年 8 月
データ・リカバリーと高可用性 ガイドおよびリファレンス	SC88-5904-00	入手可能	2009 年 8 月
データベース: 管理の概念および構成リファレンス	SC88-5870-00	入手可能	2009 年 8 月
データベースのモニタリング ガイドおよびリファレンス	SC88-5872-00	入手可能	2009 年 8 月
データベース・セキュリティー・ガイド	SC88-5905-00	入手可能	2009 年 8 月

表 1345. DB2 の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
DB2 Text Search ガイド	SC88-5902-00	入手可能	2009 年 8 月
ADO.NET および OLE DB アプリケーションの開発	SC88-5874-00	入手可能	2009 年 8 月
組み込み SQL アプリケーションの開発	SC88-5875-00	入手可能	2009 年 8 月
Java アプリケーションの開発	SC88-5878-00	入手可能	2009 年 8 月
Perl、PHP、Python および Ruby on Rails アプリケーションの開発	SC88-5879-00	入手不可	2009 年 8 月
SQL および外部ルーチンの開発	SC88-5876-00	入手可能	2009 年 8 月
データベース・アプリケーション開発の基礎	GI88-4201-00	入手可能	2009 年 8 月
DB2 インストールおよび管理 概説 (Linux および Windows 版)	GI88-4202-00	入手可能	2009 年 8 月
グローバリゼーション・ガイド	SC88-5906-00	入手可能	2009 年 8 月
DB2 サーバー機能 インストール	GC88-5888-00	入手可能	2009 年 8 月
IBM データ・サーバー・クライアント機能 インストール	GC88-5889-00	入手不可	2009 年 8 月
メッセージ・リファレンス 第 1 巻	SC88-5897-00	入手不可	2009 年 8 月
メッセージ・リファレンス 第 2 巻	SC88-5898-00	入手不可	2009 年 8 月
Net Search Extender 管理およびユーザーズ・ガイド	SC88-5901-00	入手不可	2009 年 8 月
パーティションおよびクラスタリングのガイド	SC88-5907-00	入手可能	2009 年 8 月
pureXML ガイド	SC88-5895-00	入手可能	2009 年 8 月
Query Patroller 管理およびユーザーズ・ガイド	SC88-5908-00	入手不可	2009 年 8 月

表 1345. DB2 の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
Spatial Extender および Geodetic Data Management Feature ユーザーズ・ガイドおよびリファレンス	SC88-5900-00	入手不可	2009 年 8 月
SQL プロシージャー言語: アプリケーションのイネーブルメントおよびサポート	SC88-5877-00	入手可能	2009 年 8 月
SQL リファレンス 第 1 巻	SC88-5881-00	入手可能	2009 年 8 月
SQL リファレンス 第 2 巻	SC88-5882-00	入手可能	2009 年 8 月
問題判別およびデータベース・パフォーマンスのチューニング	SC88-5871-00	入手可能	2009 年 8 月
DB2 バージョン 9.7 へのアップグレード	SC88-5887-00	入手可能	2009 年 8 月
Visual Explain チュートリアル	SC88-5899-00	入手不可	2009 年 8 月
DB2 バージョン 9.7 の新機能	SC88-5893-00	入手可能	2009 年 8 月
ワークロード・マネージャー ガイドおよびリファレンス	SC88-5894-00	入手可能	2009 年 8 月
XQuery リファレンス	SC88-5896-00	入手不可	2009 年 8 月

表 1346. DB2 Connect 固有の技術情報

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
DB2 Connect Personal Edition インストールおよび構成	SC88-5891-00	入手可能	2009 年 8 月
DB2 Connect サーバー機能 インストールおよび構成	SC88-5892-00	入手可能	2009 年 8 月
DB2 Connect ユーザーズ・ガイド	SC88-5890-00	入手可能	2009 年 8 月

表 1347. Information Integration の技術情報

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
Information Integration: Administration Guide for Federated Systems	SC19-1020-02	入手可能	2009 年 8 月
Information Integration: ASNCLP Program Reference for Replication and Event Publishing	SC19-1018-04	入手可能	2009 年 8 月
Information Integration: Configuration Guide for Federated Data Sources	SC19-1034-02	入手不可	2009 年 8 月
Information Integration: SQL Replication Guide and Reference	SC19-1030-02	入手可能	2009 年 8 月
Information Integration: Introduction to Replication and Event Publishing	GC19-1028-02	入手可能	2009 年 8 月

DB2 の印刷資料の注文方法

DB2 の印刷資料が必要な場合、オンラインで購入することができますが、すべての国および地域で購入できるわけではありません。 DB2 の印刷資料については、IBM 営業担当員にお問い合わせください。 *DB2 PDF ドキュメンテーション* DVD の一部のソフトコピー・ブックは、印刷資料では入手できないことに留意してください。例えば、「*DB2 メッセージ・リファレンス*」はどちらの巻も印刷資料としては入手できません。

DB2 PDF ドキュメンテーション DVD で利用できる DB2 の印刷資料の大半は、IBM に有償で注文することができます。国または地域によっては、資料を IBM Publications Center からオンラインで注文することもできます。お客様の国または地域でオンライン注文が利用できない場合、DB2 の印刷資料については、IBM 営業担当員にお問い合わせください。 DB2 PDF ドキュメンテーション DVD に収録されている資料の中には、印刷資料として提供されていないものもあります。

注: 最新で完全な DB2 資料は、DB2 インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7) で参照することができます。

DB2 の印刷資料は以下の方法で注文することができます。

- 日本 IBM 発行のマニュアルはインターネット経由でご購入いただけます。詳しくは http://www.ibm.com/shop/publications/order をご覧ください。資料の注文情報にアクセスするには、お客様の国、地域、または言語を選択してください。その後、各ロケーションにおける注文についての指示に従ってください。
- DB2 の印刷資料を IBM 営業担当員に注文するには、以下のようにします。

1. 以下の Web サイトのいずれかから、営業担当員の連絡先情報を見つけてください。
   - IBM Directory of world wide contacts (www.ibm.com/planetwide)
   - IBM Publications Web サイト (http://www.ibm.com/shop/publications/order)。国、地域、または言語を選択し、お客様の所在地に該当する Publications ホーム・ページにアクセスしてください。このページから、「このサイトについて」のリンクにアクセスしてください。
2. 電話をご利用の場合は、DB2 資料の注文であることをご指定ください。
3. 担当者に、注文する資料のタイトルと資料番号をお伝えください。タイトルと資料番号は、 850 ページの『DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)』でご確認いただけます。

コマンド行プロセッサーから SQL 状態ヘルプを表示する

DB2 製品は、SQL ステートメントの結果の原因になったと考えられる条件の SQLSTATE 値を戻します。 SQLSTATE ヘルプは、SQL 状態および SQL 状態クラス・コードの意味を説明します。

SQL 状態ヘルプを開始するには、コマンド行プロセッサーを開いて以下のように入力します。

```
? sqlstate or ? class code
```

ここで、*sqlstate* は有効な 5 桁の SQL 状態を、*class code* は SQL 状態の最初の 2 桁を表します。
例えば、`? 08003` を指定すると SQL 状態 08003 のヘルプが表示され、 `? 08` を指定するとクラス・コード 08 のヘルプが表示されます。

異なるバージョンの DB2 インフォメーション・センターへのアクセス

DB2 バージョン 9.7 のトピックを扱っている DB2 インフォメーション・センターの URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/です。

DB2 バージョン 9.5 のトピックを扱っている DB2 インフォメーション・センターの URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/です。

DB2 バージョン 9 のトピックを扱っている DB2 インフォメーション・センターの URL は http://publib.boulder.ibm.com/infocenter/db2luw/v9/です。

DB2 バージョン 8 のトピックについては、バージョン 8 のインフォメーション・センターの URL http://publib.boulder.ibm.com/infocenter/db2luw/v8/にアクセスしてください。

DB2 インフォメーション・センターでの希望する言語でのトピックの表示

DB2 インフォメーション・センターでは、ブラウザーの設定で指定した言語でのトピックの表示が試みられます。トピックがその指定言語に翻訳されていない場合は、 DB2 インフォメーション・センターでは英語でトピックが表示されます。

- Internet Explorer Web ブラウザーで、指定どおりの言語でトピックを表示するには、以下のようにします。
  1. Internet Explorer の「**ツール**」 -> 「**インターネット オプション**」 -> 「**言語...**」ボタンをクリックします。「言語の優先順位」ウィンドウがオープンします。
  2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
     - リストに新しい言語を追加するには、「**追加...**」ボタンをクリックします。

       注: 言語を追加しても、特定の言語でトピックを表示するのに必要なフォントがコンピューターに備えられているとはかぎりません。
     - リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで「**上に移動**」ボタンをクリックします。
  3. ブラウザー・キャッシュを消去してから、ページを最新表示します。希望する言語で DB2 インフォメーション・センターが表示されます。
- Firefox または Mozilla Web ブラウザーの場合に、希望する言語でトピックを表示するには、以下のようにします。
  1. 「**ツール**」 -> 「**オプション**」 -> 「**詳細**」 ダイアログの「**言語**」セクションにあるボタンを選択します。「設定」ウィンドウに「言語」パネルが表示されます。
  2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
     - リストに新しい言語を追加するには、「**追加...**」ボタンをクリックしてから、「言語を追加」ウィンドウで言語を選択します。
     - リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで「**上に移動**」ボタンをクリックします。
  3. ブラウザー・キャッシュを消去してから、ページを最新表示します。希望する言語で DB2 インフォメーション・センターが表示されます。

ブラウザーとオペレーティング・システムの組み合わせによっては、オペレーティング・システムの地域の設定も希望のロケールと言語に変更しなければなりません。

コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの更新

ローカルにインストールされた DB2 インフォメーション・センターは、定期的に更新する必要があります。

始める前に

DB2 バージョン 9.7 インフォメーション・センターが既にインストールされている必要があります。詳しくは、「*DB2 サーバー機能 インストール*」の『DB2 セットアップ・ウィザードによる DB2 インフォメーション・センターのインストール』

のトピックを参照してください。インフォメーション・センターのインストールに適用されるすべての前提条件と制約事項は、インフォメーション・センターの更新にも適用されます。

このタスクについて

既存の DB2 インフォメーション・センターは、自動で更新することも。手動で更新することもできます。

- 自動更新 - 既存のインフォメーション・センターのフィーチャーと言語を更新します。自動更新を使用すると、更新中にインフォメーション・センターが使用できなくなる時間が最小限で済むというメリットもあります。さらに、自動更新は、定期的に実行する他のバッチ・ジョブの一部として実行されるように設定することができます。
- 手動更新 - 更新処理中にフィーチャーまたは言語を追加する場合に使用する必要があります。例えば、ローカルのインフォメーション・センターが最初は英語とフランス語でインストールされており、その後ドイツ語もインストールすることにした場合、手動更新でドイツ語をインストールし、同時に、既存のインフォメーション・センターのフィーチャーおよび言語を更新できます。しかし、手動更新ではインフォメーション・センターを手動で停止、更新、再始動する必要があります。更新処理の間はずっと、インフォメーション・センターは使用できなくなります。

手順

このトピックでは、自動更新のプロセスを詳しく説明しています。手動更新の手順については、『コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの手動更新』のトピックを参照してください。

コンピューターまたはイントラネット・サーバーにインストールされている DB2 インフォメーション・センターを自動で更新するには、次のようにします。

1. Linux オペレーティング・システムの場合、次のようにします。
   a. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、/opt/ibm/db2ic/V9.7 ディレクトリーにインストールされています。
   b. インストール・ディレクトリーから doc/bin ディレクトリーにナビゲートします。
   c. 次のように ic-update スクリプトを実行します。

      ```
      ic-update
      ```
2. Windows オペレーティング・システムの場合、次のようにします。
   a. コマンド・ウィンドウを開きます。
   b. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、<Program Files>¥IBM¥DB2 Information Center¥Version 9.7 ディレクトリーにインストールされています (<Program Files> は「Program Files」ディレクトリーのロケーション)。

c. インストール・ディレクトリーから doc¥bin ディレクトリーにナビゲートします。

d. 次のように ic-update.bat ファイルを実行します。

```
ic-update.bat
```

結果

DB2 インフォメーション・センターが自動的に再始動します。更新が入手可能な場合、インフォメーション・センターに、更新された新しいトピックが表示されます。インフォメーション・センターの更新が入手可能でなかった場合、メッセージがログに追加されます。ログ・ファイルは、doc¥eclipse¥configuration ディレクトリーにあります。ログ・ファイル名はランダムに生成された名前です。例えば、1239053440785.log のようになります。

コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの手動更新

DB2 インフォメーション・センターをローカルにインストールしている場合は、IBM から資料の更新を入手してインストールすることができます。

ローカルにインストールされた DB2 インフォメーション・センターを手動で更新するには、以下のことを行う必要があります。

1. コンピューター上の DB2 インフォメーション・センターを停止し、インフォメーション・センターをスタンドアロン・モードで再始動します。インフォメーション・センターをスタンドアロン・モードで実行すると、ネットワーク上の他のユーザーがそのインフォメーション・センターにアクセスできなくなります。これで、更新を適用できるようになります。DB2 インフォメーション・センターのワークステーション・バージョンは、常にスタンドアロン・モードで実行されます。を参照してください。
2. 「更新」機能を使用することにより、どんな更新が利用できるかを確認します。インストールしなければならない更新がある場合は、「更新」機能を使用してそれを入手およびインストールできます。

   注: ご使用の環境において、インターネットに接続されていないマシンに DB2 インフォメーション・センターの更新をインストールする必要がある場合、インターネットに接続されていて DB2 インフォメーション・センターがインストールされているマシンを使用して、更新サイトをローカル・ファイル・システムにミラーリングしてください。ネットワーク上の多数のユーザーが資料の更新をインストールする場合にも、更新サイトをローカルにミラーリングして、更新サイト用のプロキシーを作成することにより、個々のユーザーが更新を実行するのに要する時間を短縮できます。
   更新パッケージが入手可能な場合、「更新」機能を使用してパッケージを入手します。ただし、「更新」機能は、スタンドアロン・モードでのみ使用できます。
3. スタンドアロンのインフォメーション・センターを停止し、コンピューター上の DB2 インフォメーション・センターを再開します。

注: Windows 2008、Windows Vista (およびそれ以上) では、このセクションの後の部分でリストされているコマンドは管理者として実行する必要があります。完全な

管理者特権でコマンド・プロンプトまたはグラフィカル・ツールを開くには、ショートカットを右クリックしてから、 「**管理者として実行**」を選択します。

コンピューターまたはイントラネット・サーバーにインストール済みの DB2 インフォメーション・センターを更新するには、以下のようにします。

1. DB2 インフォメーション・センターを停止します。
   - Windows では、「**スタート**」 → 「**コントロール パネル**」 → 「**管理ツール**」 → 「**サービス**」をクリックします。次に、「**DB2 インフォメーション・センター**」サービスを右クリックして「**停止**」を選択します。
   - Linux では、以下のコマンドを入力します。

     ```
     /etc/init.d/db2icdv97 stop
     ```
2. インフォメーション・センターをスタンドアロン・モードで開始します。
   - Windows の場合:
     a. コマンド・ウィンドウを開きます。
     b. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、<Program Files>¥IBM¥DB2 Information Center¥Version 9.7 ディレクトリーにインストールされています (<Program Files> は「Program Files」ディレクトリーのロケーション)。
     c. インストール・ディレクトリーから doc¥bin ディレクトリーにナビゲートします。
     d. 次のように help_start.bat ファイルを実行します。

        ```
        help_start.bat
        ```
   - Linux の場合:
     a. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、/opt/ibm/db2ic/V9.7 ディレクトリーにインストールされています。
     b. インストール・ディレクトリーから doc/bin ディレクトリーにナビゲートします。
     c. 次のように help_start スクリプトを実行します。

        ```
        help_start
        ```

   システムのデフォルト Web ブラウザーが開き、スタンドアロンのインフォメーション・センターが表示されます。
3. 「**更新**」ボタン ( ) をクリックします。(ブラウザーで JavaScript™ が有効になっている必要があります。) インフォメーション・センターの右側のパネルで、「**更新の検索 (Find Updates)**」をクリックします。 既存の文書に対する更新のリストが表示されます。
4. インストール・プロセスを開始するには、インストールする更新をチェックして選択し、「**更新のインストール**」をクリックします。
5. インストール・プロセスが完了したら、「**完了**」をクリックします。
6. 次のようにして、スタンドアロンのインフォメーション・センターを停止します。

- Windows の場合は、インストール・ディレクトリーの doc¥bin ディレクトリーにナビゲートしてから、次のように help_end.bat ファイルを実行します。

  ```
  help_end.bat
  ```

  注: help_end バッチ・ファイルには、help_start バッチ・ファイルを使用して開始したプロセスを安全に停止するのに必要なコマンドが含まれています。help_start.bat は、Ctrl-C や他の方法を使用して停止しないでください。
- Linux の場合は、インストール・ディレクトリーの doc/bin ディレクトリーにナビゲートしてから、次のように help_end スクリプトを実行します。

  ```
  help_end
  ```

  注: help_end スクリプトには、help_start スクリプトを使用して開始したプロセスを安全に停止するのに必要なコマンドが含まれています。他の方法を使用して、help_start スクリプトを停止しないでください。

7. DB2 インフォメーション・センターを再開します。
   - Windows では、「**スタート**」 → 「**コントロール パネル**」 → 「**管理ツール**」 → 「**サービス**」をクリックします。次に、「**DB2 インフォメーション・センター**」サービスを右クリックして「**開始**」を選択します。
   - Linux では、以下のコマンドを入力します。

     ```
     /etc/init.d/db2icdv97 start
     ```

更新された DB2 インフォメーション・センターに、更新された新しいトピックが表示されます。

DB2 チュートリアル

DB2 チュートリアルは、DB2 製品のさまざまな機能について学習するのを支援します。この演習をとおして段階的に学習することができます。

はじめに

インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2help/) から、このチュートリアルの XHTML 版を表示できます。

演習の中で、サンプル・データまたはサンプル・コードを使用する場合があります。個々のタスクの前提条件については、チュートリアルを参照してください。

DB2 チュートリアル

チュートリアルを表示するには、タイトルをクリックします。

「*pureXML ガイド*」の『**pureXML™**』
: XML データを保管し、ネイティブ XML データ・ストアに対して基本的な操作を実行できるように、DB2 データベースをセットアップします。

「*Visual Explain チュートリアル*」の『**Visual Explain**』
: Visual Explain を使用して、パフォーマンスを向上させるために SQL ステートメントを分析し、最適化し、調整します。

DB2 トラブルシューティング情報

DB2 データベース製品を使用する際に役立つ、トラブルシューティングおよび問題判別に関する広範囲な情報を利用できます。

DB2 ドキュメンテーション

トラブルシューティング情報は、「*DB2 問題判別ガイド*」、または*DB2 インフォメーション・センター*の『データベースの基本』セクションにあります。ここには、DB2 診断ツールおよびユーティリティーを使用して、問題を切り分けて識別する方法、最も頻繁に起こる幾つかの問題に対するソリューションについての情報、および DB2 データベース製品を使用する際に発生する可能性のある問題の解決方法についての他のアドバイスがあります。

DB2 Technical Support の Web サイト

現在問題が発生していて、考えられる原因とソリューションを検索したい場合は、DB2 Technical Support の Web サイトを参照してください。Technical Support サイトには、最新の DB2 資料、TechNotes、プログラム診断依頼書 (APAR またはバグ修正)、フィックスパック、およびその他のリソースへのリンクが用意されています。この知識ベースを活用して、問題に対する有効なソリューションを探し出すことができます。

DB2 Technical Support の Web サイト (http://www.ibm.com/software/data/db2/support/db2_9/) にアクセスしてください。

ご利用条件

これらの資料は、以下の条件に同意していただける場合に限りご使用いただけます。

個人使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

商業的使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。 ただし、IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態で提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。

付録 B. 特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。IBM 以外の製品に関する情報は、本書の最初の発行時点で入手可能な情報に基づいており、変更される場合があります。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-8711
東京都港区六本木 3-2-12
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを

経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。サンプル・プログラムは、現存するままの状態で提供されるものであり、いかなる種類の保証も提供されません。 IBM は、これらのサンプル・プログラムの使用から生ずるいかなる損害に対しても責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

© (*お客様の会社名*) (*西暦年*). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _*年を入れる*_. All rights reserved.

商標

IBM、IBM ロゴおよび ibm.com® は、世界の多くの国で登録された International Business Machines Corp. の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、www.ibm.com/legal/copytrade.shtml の「Copyright and trademark information」をご覧ください。

以下は、それぞれ各社の商標または登録商標です。

- Linux は、Linus Torvalds の米国およびその他の国における商標です。
- Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標です。
- UNIX は、The Open Group の米国およびその他の国における登録商標です。
- Intel、Intel ロゴ、Intel Inside®、 Intel Inside ロゴ、Intel® Centrino®、Intel Centrino ロゴ、Celeron®、Intel® Xeon®、Intel SpeedStep®、Itanium®、Pentium® は、Intel Corporation または子会社の米国およびその他の国における商標または登録商標です。
- Microsoft、 Windows、 Windows NT®、および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

［サ行］

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

［ワ行］

A

B

C

D

F

G

I

J

L

M

N

O

P

R

S

T

U

V

W

X

[特殊文字]

Printed in Japan

SC88-5872-00

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12

Spine information:

IBM DB2 9.7 for Linux, UNIX, and Windows

データベースのモニタリング　ガイドおよびリファレンス