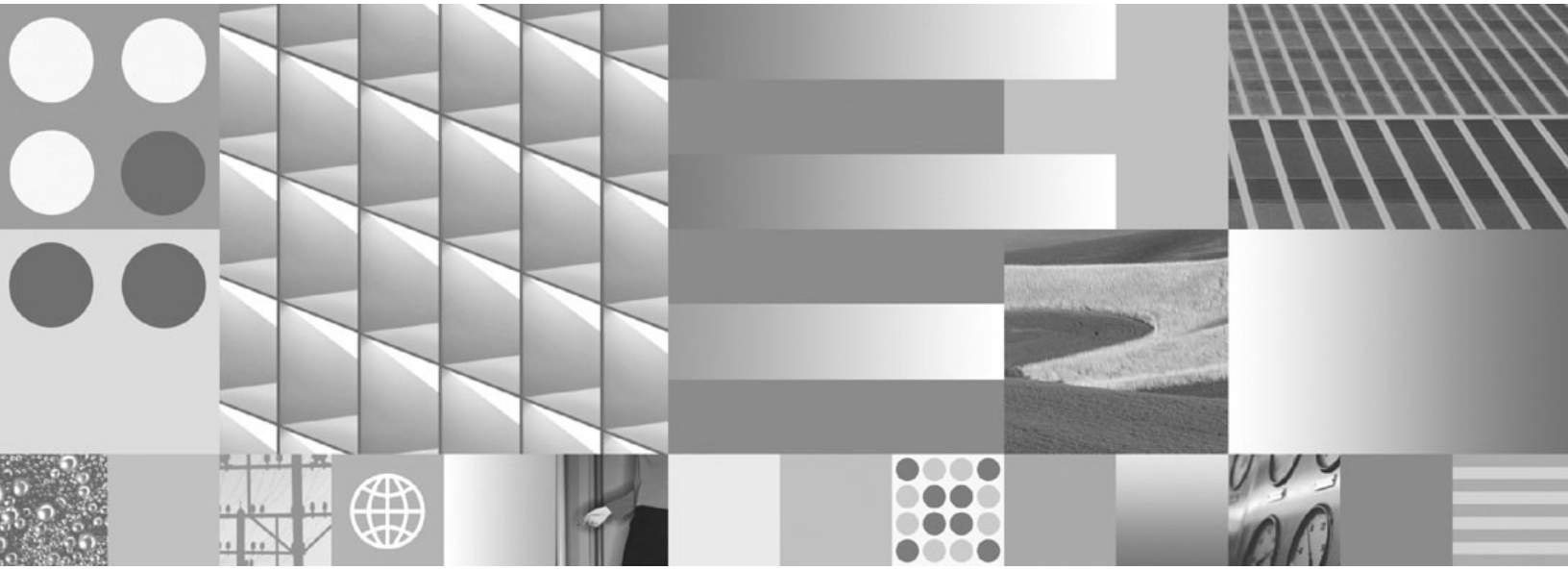
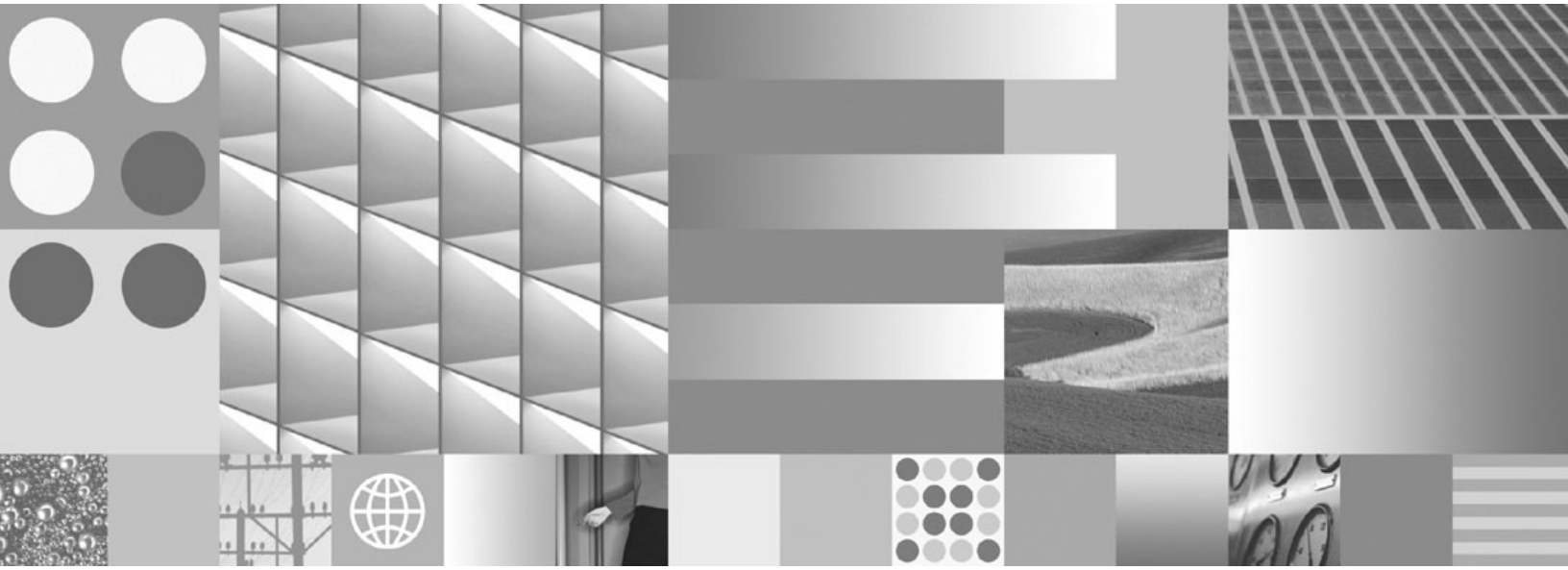
DB2 Version 9.5
for Linux, UNIX, and Windows



ワークロード・マネージャー ガイドおよびリファレンス

SC88-4446-01
(英文原典：SC23-5870-01)

DB2 Version 9.5
for Linux, UNIX, and Windows



ワークロード・マネージャー ガイドおよびリファレンス

SC88-4446-01
(英文原典：SC23-5870-01)

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、 345 ページの『付録 H. 特記事項』に記載されている情報をお読みください。

当版に関する特記事項

本書には、IBM の専有情報が含まれています。その情報は、使用許諾条件に基づき提供され、著作権により保護されています。本書に記載される情報には、いかなる製品の保証も含まれていません。また、本書で提供されるいかなる記述も、製品保証として解釈すべきではありません。

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。

IBM 発行のマニュアルに関する情報のページ

`http://www.ibm.com/jp/manuals/`

こちらから、日本語版および英語版のオンライン・ライブラリーをご利用いただけます。また、マニュアルに関するご意見やご感想を、上記ページよりお送りください。今後の参考にさせていただきます。

(URL は、変更になる場合があります)

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： SC23-5870-01
DB2 Version 9.5 for Linux, UNIX, and Windows
Workload Manager Guide and Reference

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2008.3

© Copyright International Business Machines Corporation 2007, 2008. All rights reserved.

目次

© Copyright IBM Corp. 2007, 2008

本書について

本書には、業務目的に適った堅固かつ予測可能な実行環境を実現するために使用できる DB2® ワークロード管理フィーチャーと機能についての情報があります。ワークロード管理を行うことにより、要求とリソースの両方が管理されます。本書では、データ・サーバー上のワークロードについてのモニター、およびトラブルシューティングの実行に関する情報を提供します。

© Copyright IBM Corp. 2007, 2008

第 1 部 概要

© Copyright IBM Corp. 2007, 2008

第 1 章 ワークロード管理の概念の紹介

効果的なワークロード管理システムは、作業が発生する環境において、目標を効率的に達成する上で役立ちます。効果的なワークロード管理システムの必要性を示す例は、枚挙にいとまがありません。

例えば、食料品店について考えてみましょう。そこでは、顧客へのサービス提供、棚への陳列、在庫管理などのさまざまな活動、つまりアクティビティーを考慮する必要があります。そして、シンプルな目標も設定するでしょう。店主は、店内を歩き回る顧客数と顧客が購入する品数の両方を最大にするという 2 つの目標を達成すると同時に、顧客が満足し、再度来店したいという願いを持って店舗から出ることを願っています。店主は、買い物をする顧客のために十分な量の在庫が必ずあるようにする必要もあります (ただし無駄が発生すると問題となるため、在庫は過剰にならないようにします)。店主はさらに、顧客が購入したものを追跡記録し、この情報を使って、顧客が再度来店するよう促すことを意図した広告を作成します。モニター機構で在庫を追跡し、在庫が少なくなったら通知を送信するようにします。万引きを検出するためにセキュリティー装置を設置します。数品目しか購入しない買い物客が、たくさんの品目を購入する他の顧客の後ろで待たずに購入できるように、特別の優先レジを作ります。これらすべての目標が達成され、これらすべての運用手順が適切に機能するなら、顧客は満足し、別の店舗に行かずにまた来店するでしょう。これらの目標と運用手順はすべて、ワークロード管理の一環として行われます。

データ・サーバー環境では、作業をさらに効果的に管理する必要があります。データ・サーバーがかつてないほど重視されている現在においてはなおさらそのようにいえます。何千というデータ挿入がレジで生成されます。売上目標が達成されているかを調べるためにレポートが常に生成され、収集データをロードするためにバッチ・アプリケーションが実行され、さらにデータを保護してサーバーを最適な状態で実行させるためにバックアップと再編成などの管理タスクが実行されます。こうしたすべての操作はすべて同じデータベース・システムを使用していて、なおかつ同じリソースを獲得するために競合しています。

データ・サーバーを実行するための目標を確実に達成するためには、効率的なワークロード管理システムがどうしても求められます。

ワークロード管理のステージ

ワークロード管理には、はっきり定義された次の 3 つのステージがあります。つまり、データ・サーバーに存在する作業の識別、実行中の作業の管理、およびデータ・サーバーが効率的に使用されていることを確認するためのモニターというステージです。

ワークロード管理を適切に行うには、目標を明確にすることから始め、多くの側面を考慮する必要があります。 『第 1 章 ワークロード管理の概念の紹介』で説明されている食料品店を例に取ると、目標には、顧客の購入金額を最大にすること、万引きを最小にすること、再度来店してもらえるように顧客が満足して店を出るようにすることが含まれるかもしれません。

© IBM Corporation 2007, 2008

データ・サーバー環境でも目標を定義する必要があります。目標は明確なものである場合もあります。サービス・レベル契約 (SLA) を目的として考え出された目標であれば特にそういえます。例えば、特定のアプリケーションからの照会が消費できる量を、CPU 合計リソースの 10% を超えないようにします。また目標を、特定の時刻に関連付けることができます。例えば、毎日の販売レポートが時間どおりに作成されるように、夜間バッチ・ユーティリティーは、データのロードを午前 8 時までに完了する必要があるかもしれません。他の状況では、目標を定量化することが難しい場合もあります。目標は、データベースのユーザーが満足し、異常なデータベース・アクティビティーが発生してユーザーの日常の作業を妨げることがないようにする、というものかもしれません。目標を定量化できるものであってもそうでなくても、ワークロード管理についての以下の各ステージを考慮する際に目標を明確にすることは非常に重要です。

- 識別。何らかの作業の目標を達成しようとする場合、まず作業に関する詳細を識別できなければなりません。食料品店では、買い物客情報はクレジット・カードとデビット・カードから識別できますし、未払いの品目はその品目に付いているアクティブなセキュリティー・タグから識別できます。データ・サーバーの場合には、システムに存在する作業を識別する方法を決定する必要があります。その作業をサブミットするアプリケーションの名前や許可 ID、またはある種の ID を提供する要素を組み合わせて使用することができます。
- 管理。管理フェーズには、目標に向かって着実に進行していくメカニズムと、目標を達成できなかった場合に取るアクションが含まれます。メカニズムの例には、優先レジでの価格確認管理があります。優先レジを設置するとスループットが早くなり顧客は満足するはずですが、牛乳のカートンに間違った値段が付いていて価格確認が必要になった場合、優先レジの流れは低下してしまいます。その問題の管理方法として、可能ならば別のレジを開き、迅速に価格確認をし、こうしたことが再度起きないように価格の問題を解決するようにします。データ・サーバー上では、記述が悪い SQL ステートメントがいくつか実行されていたり、ピーク時にボリュームが急激に多くなったり、同じリソースに対して異なるアプリケーションによる競合があまりにも多かったりして、全体的なパフォーマンスが悪いことに気付くかもしれません。管理フェーズには、目標を達成するためにリソースを割り当てるメカニズムと、目標を達成できなかった場合にとるアクションが含まれます。
- モニター。モニターが重要である理由は、いくつかあります。第 1 に、目標を達成しているかを調べるには、その目標への進行状況を追跡するメカニズムが必要です。さらに、モニターを行うと、目標の達成を妨げている可能性がある問題を識別する上で役立ちます。店舗においては、店主は顧客の流れを監視したり、万引き、特定の販売品目の在庫が危険なレベルまで不足した状態などの問題の警告を自動的に受け取るようにしたり、店舗での製品の最適な配置方法を判断するために過去の消費行動パターンを分析したりすることができます。データ・サーバーにおいては、データベース・アクティビティーの応答時間には通常明示的な目標があるため、この測定基準について測定するための方法があることと傾向を監視することは重要です。

以下の図は、ワークロード管理のステージを示しています。

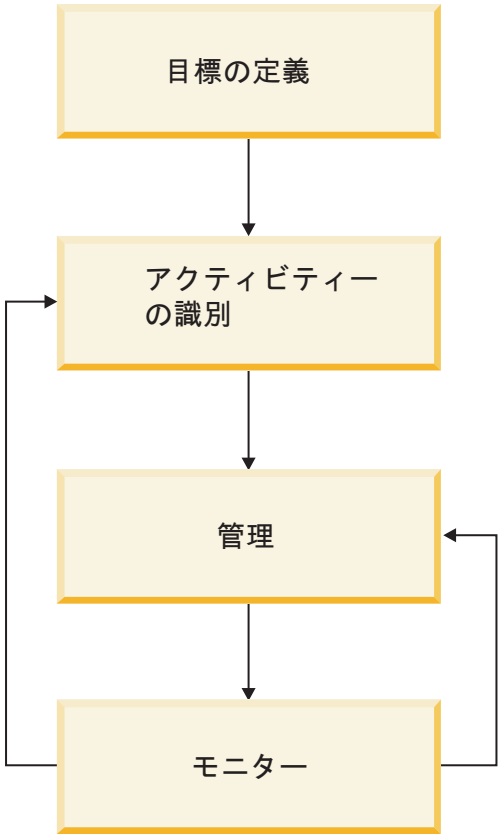


図 1. ワークロード管理のステージ

ワークロード管理の識別ステージ

ワークロード管理ソリューションを実施するための最初の段階は、データ・サーバー上で実行される作業の識別です。

データベース・アクティビティーの識別には、幾つかの方法を使用できます。例えば、ソース (つまり作業をサブミットした人) ごとにアクティビティーを識別できます。以下の図は、異なるユーザー、グループ、およびアプリケーションに由来する、様々な作業ソースを示しています。またアクティビティーを、タイプ別に識別することもできます。

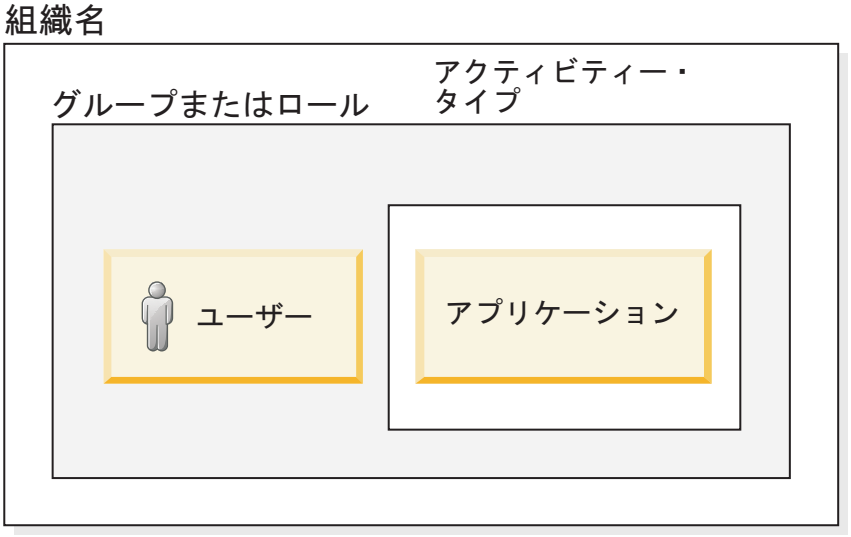


図 2. データ・サーバー上のデータベース・アクティビティーのさまざまなソースとタイプ

識別を実施するために、ワークロード・オブジェクトとワーク・クラス・オブジェクトという 2 つの新しいデータベース・オブジェクトが提供されています。

ワークロード

ワークロード は、サブミットされた作業を後で管理できるように、そのソースに基づいてそれを識別するために使用されるオブジェクトです。そのソースは、作業がサブミットされたデータベース接続の属性を使用して決定されます。

接続の属性は接続が確立された時に評価されます。また接続はワークロードの新しいオカレンス (ワークロード・オカレンスという) が開始する時点でワークロード定義に割り当てられます。接続の属性のいずれかがその接続の存続期間中に変更される場合、そのワークロードの割り当ては変更後の次の作業単位が開始する時に再評価されます。新規のワークロード定義が割り当てられる場合、以前に割り当てられたワークロードの古いワークロード・オカレンスは終了し、新たに割り当てられたワークロード定義の新規のオカレンスが開始します。それぞれの接続は、どんな場合でも 1 つのワークロードだけに割り当てられますが、複数の接続が同じワークロードに同時に割り当てられ、その定義に関連した複数のワークロード・オカレンスが並行して存在する結果となる可能性もあります。

ワークロードのサポートされている接続の属性については、 45 ページの『第 3 章 ワークロード』を参照してください。

接続属性の値またはワークロード定義自体が作業単位の間に変更される場合には、各作業単位の初めにワークロードの再評価が行われます。この再評価の結果として、接続が新しいワークロードに関連付けられ、別のワークロード・オカレンスが作成される場合があります。

ワークロードの作成の最初の例

```
CREATE WORKLOAD "REPORTING" APPLNAME('Accounts') SERVICE CLASS Marketing
```

このコマンドにより、REPORTING というワークロード・オブジェクトが作成されます。 Accounts というアプリケーション名の属性を持つすべての接続はこのワー

クロードに割り当てられ、以下の図に示されているように、それらは Marketing サービス・クラスで実行されます。

図 3. REPORTING ワークロード

ワークロードの作成の 2 番目の例

```
CREATE WORKLOAD "SUMMARY" SESSION_USER_GROUP('Deptmgr') APPLNAME('Accounts')
SERVICE CLASS HumanResources
```

このコマンドにより、SUMMARY というワークロード・オブジェクトが作成されます。アプリケーション名が Accounts で、セッション・ユーザー・グループ Deptmgr (そのセッション・ユーザーの許可 ID は DeptMgr グループに所属する) に所属するすべての接続は SUMMARY ワークロードにマップされ、以下の図に示されているように、それらは HumanResources サービス・クラスで実行されるように割り当てられます。

図 4. SUMMARY ワークロード

ワークロードの評価中にカスタム定義のワークロードに割り当てられない接続は、デフォルトのワークロードに割り当てられます。デフォルトのワークロードは、すべてのデータベース接続がワークロードに必ず関連付けられるようにします。

ワーク・クラス

アクティビティーのソースに注目するデータベース接続属性の使用に加えて、オプションのワーク・クラス を作成すると、作業のタイプに基づいてアクティビティーを識別できます。

ワーク・クラス・セット は、ワーク・アクション・セットとともに DB2 データ・サーバーのさまざまなパーツで使用することができる作業タイプの共通の定義を示しています。

表 1. 作業タイプ

作業タイプ	説明
READ	データのフェッチのみが行われる (つまり表は更新されない) すべての SELECT および XQuery ステートメントを含めます。
WRITE	データ・サーバー上のデータ内容を変更するすべてのステートメント (つまり、INSERT、UPDATE、DELETE、および MERGE。これらが SELECT ステートメントに組み込まれている場合も含む) を含めます。

表 1. 作業タイプ (続き)

作業タイプ	説明
CALL	CALL ステートメントを使用する、プロシージャーのすべての呼び出しを含めます。
DML	READ および WRITE 作業タイプにみられる作業を組み合わせます。
DDL	データベース・オブジェクトを作成または変更するステートメント (つまり、CREATE、ALTER、DROP、COMMENT、DECLARE GLOBAL TEMPORARY TABLE、REFRESH TABLE、RENAME、GRANT、REVOKE、SET INTEGRITY) を含めます。
LOAD	データ・サーバー上でロード・ユーティリティーにより開始されたすべての作業を含めます。
ALL	すべてのタイプの作業を含めます。

ワーク・クラスでは、DML 作業 (あるいは READ および WRITE ステートメント) の ID に予測エレメントを使用する機能も取り入れています。予測エレメントは非常に役に立ちます。なぜなら、データベース・アクティビティーがデータ・サーバーのリソースの消費を開始する前にアクションを取るために使用できる、それらのアクティビティーに関する情報を得られるからです。以下の表には、予測エレメントに関する情報が示されています。

表 2. 予測識別の特性

予測エレメント	説明
見積コスト	特定の timeron の範囲内の DML を含めるため、DB2 コンパイラーから取得できる見積コストを使用します (例えば、1,000,000 timeron を超える見積コストを持つすべての大規模な照会に対してワーク・クラス・セットを作成します)
見積カーディナリティー	特定の戻り行数の範囲内の DML を含めるため、DB2 コンパイラーから返される見積行数 (カーディナリティー) を使用します (例えば、500,000 行を超える行を返すと見積もられる大規模な照会に対してワーク・クラスを作成します)

CALL ステートメントが呼び出すプロシージャーのスキーマ名を使用することによりアクティビティーを識別することもできます。ワークロード属性とワーク・クラス・タイプに基づいて作業を識別して、次のステージである作業の管理のために準備できます。

ワーク・アクションを使って、ワーク・クラスでアクティビティーを管理する方法の一部を定義することもできます。ワーク・クラスをアクティブにするには、そのために少なくとも 1 つのワーク・アクションを定義する必要があります。詳しくは、9 ページの『ワークロード管理の管理ステージ』にあるワーク・アクション・セットについての説明を参照してください。

ワークロード管理の管理ステージ

作業の識別に続く次のステージは、作業のアクティブ管理です。ここではリソースを割り当て、その作業の制御を設定します。

サービス・クラス

サービス・クラスの目的は、作業を実行できる実行環境を定義することです。この環境には、使用可能なリソースとさまざまな実行しきい値を含めることができます。

ワークロードを定義する際、そのワークロードに関連付けられる作業が実行されるサービス・クラスを示す必要があります。デフォルトのワークロードも存在しているため、データ・サーバーのすべての作業は 1 つのサービス・クラスの中で必ず実行されます。

優先順位付けとリソース制御

サービス・クラス・オブジェクトを作成または変更する際、次のようないくつかのリソース制御を定義できます。

表 3. サービス・クラスにより得られるリソース制御

制御	説明
Agent priority	この制御は、サービス・クラスで実行中のエージェント・スレッドの CPU 優先順位レベルを設定します。この優先順位は、このデータ・サーバーで実行中の他のスレッドおよびプロセスに対する相対 (デルタ) 優先順位として、オペレーティング・システムに渡されます。 **注:** アウトバウンドの相関関係子の使用中はアクティブになりません。
Prefetch priority	この制御はプリフェッチ要求に対する優先順位を割り当て、プリフェッチ要求がデータ・サーバーによって扱われる順序に影響を与えます。
Outbound correlator	この制御を使用すると、ワークロードのリソースを外部ワークロード・マネージャーで制御できます (現在サポートされている外部ワークロード・マネージャーは AIX® ワークロード・マネージャーのみです)。タグは、エージェントから外部ワークロード・マネージャーに渡され、マネージャーで定義されたリソース・グループにマップされます。 DB2 ワークロード管理と AIX ワークロード・マネージャーとを併用する場合は、追加の機能が使用可能です。 AIX ワークロード・マネージャーを使用して、それぞれのサービス・クラスに割り振られる CPU の量を制御することができます。オプションには、各サービス・クラスに対する CPU の最小、最大、または相対比率の共有の設定が含まれます。

サービス・サブクラス

サービス・スーパークラスは作業における最高位の層ですが、アクティビティーはサービス・サブクラスでだけ実行されます。この区別に注意することは重要です。各サービス・スーパークラスには、明示的に定義されたサブクラスに割り当てない

アクティビティーを実行するための、デフォルトのサービス・サブクラスが定義されています。サービス・スーパークラスが作成されると、このデフォルトのサブクラスが作成されます。さらに作業またはリソースを分離することが必要であれば、それに応じてサービス・クラス内に追加のサブクラスを作成することができます。

定義できるサブクラスは単一のレベルだけです (つまり、サブクラスはサービス・スーパークラスの下位にしか定義できず、別のサブクラスの下位には定義できません)。

以下の図は、ワークロードおよびサービス・クラスを使用したカスタム・ワークロード管理構成の例です。

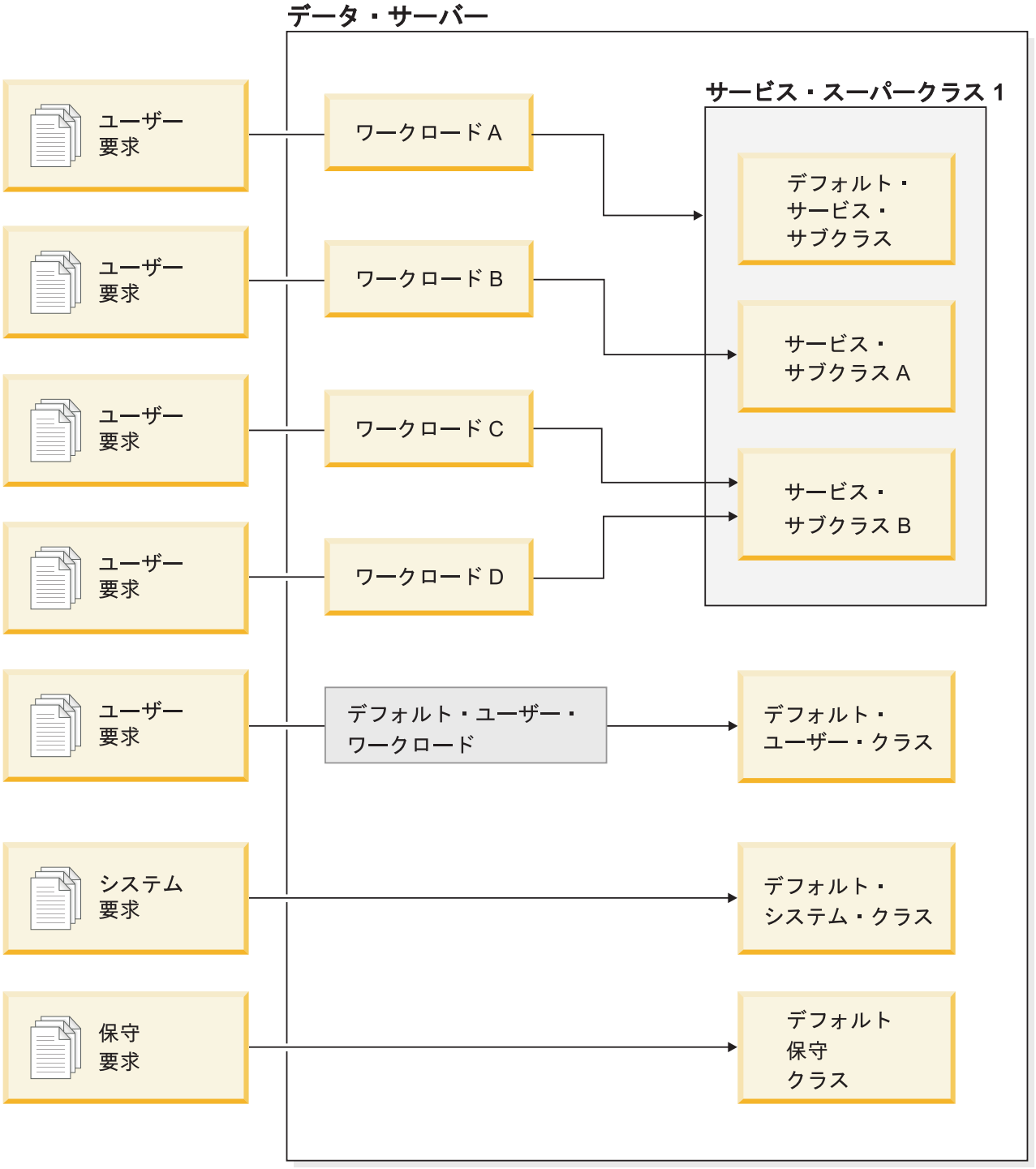


図 5. ワークロードおよびサービス・クラスを使用したカスタム・ワークロード管理構成

ユーザー要求がデータ・サーバーに入ると、それは特定のワークロードに所属するものとして識別され、サービス・スーパークラスまたはサブクラスに割り当てられます。さらに、特別なデフォルトのシステム・サービス・クラスの下で実行されるシステム要求 (例えばプリフェッチなど) や、デフォルトの保守サービス・クラスの下で実行される DB2 主導型の保守要求 (ヘルス・モニターからの自動 RUNSTATS など) もあります。

しきい値

リソース制御は、データ・サーバーの一定の健全な状態を維持するための 1 つの方法です。健全とは、定義した目標が達成されている、という意味です。時には、通常の予想を超えるような作業 (何十万という行を返す照会など) が入ってきて、システム上で実行中の他のすべての作業を犠牲にして貴重なリソースを消費してしまう場合もあります。

しきい値オブジェクトを作成して、システム内の秩序を維持したり、異常な動作をする作業をキャッチしたりします。以下の表には、定義できるさまざまなしきい値に関する情報が示されています。しきい値の最初のセットはアクティビティーの限界を扱うもので、制御は、アクティビティーがデータ・サーバーの実行方法に与える影響に関連しています。時間の超過、通常を超えて返される大量のデータ、および通常を超えて消費される大量のリソースの発生は、潜在的に問題があるアクティビティーが基準を超えてリソースを消費している可能性があることの警告標識です。

表 4. アクティビティー限界しきい値

しきい値	説明
ACTIVITYTOTALTIME	DB2 データ・サーバーで、特定のアクティビティーがサブミットされてから完了までに費やすことができる時間の長さ (実行時間およびキュー時間) を制御します。完了までに異常に長い時間がかかっているジョブを検出するために使用します。
CONNECTIONIDLETIME	接続がアイドル状態にあり、ユーザー要求のために作動していない時間の長さを制御します。データ・サーバーのリソースを効率的に使用していない状態と、アプリケーションの待ち状態を検出するために使用します。
ESTIMATEDSQLCOST	DB2 オプティマイザーによって見積コストが大きいと判断される DML アクティビティーを制御します。リソースを大量に消費する可能性がある SQL がシステムで実行開始され、不適切な書き方の SQL が識別される前にそれを予測するために使用します。
SQLROWSRETURNED	SQL を実行する時に返される行数を制御します。データ量が適切なボリュームを超えた時を識別するために使用します。
SQLTEMPSPACE	特定のアクティビティーがパーティション上で消費できる TEMPORARY 表スペースの量を制御します。いくつかの正しくない SQL ステートメントが大量の一時スペースを独占し、他の作業の進行を妨げてしまうような状態を防ぐために使用します。

しきい値の次のセットは並行性制御を扱うもので、これは、同時に実行される特定のアクティビティーがデータ・サーバーに与える影響を減らすために、それらの数を制限する必要がある状態の発生を監視するためのものです。

表 5. 並行性限界しきい値

しきい値	説明
CONCURRENTWORKLOADOCCURRENCES	コーディネーター・パーティションで同時に実行できるワークロードのアクティブなオカレンスの数を制御します。特定のソースからの接続の広がりを制御するために使用します。
TOTALDBPARTITIONCONNECTIONS	特定のパーティションに対して同時に確立できるデータベース接続の数を制御します。特定のパーティションが過負荷とならないようにするために使用します。

表 5. 並行性限界しきい値 (続き)

しきい値	説明
TOTALSCPARTITIONCONNECTIONS	特定のサービス・クラスの中で実行される作業用の特定のパーティションに対して同時に確立できるデータベース接続の数を制御します。データベース・パーティション接続の合計数に似ていますが、接続がサービス・クラスとリンクされているため、細分性は高くなります。
CONCURRENTWORKLOADACTIVITIES	1 つのワークロード・オカレンスの中で実行できる個々のアクティビティーの数を制御します。個々のワークロード・オカレンスの中での作業を制限するために使用します。
CONCURRENTDBCOORDACTIVITIES	しきい値が関連しているドメイン内の並行アクティビティー (データベース、ワーク・アクション、サービス・スーパークラス、またはサービス・サブクラス) の数を制御します。

しきい値に違反した場合に取ることができるアクションの種類は、しきい値自体によって異なります。

データの収集

一部のしきい値では、違反が生じると何らかの形式のデータが収集されます。デフォルトでは、しきい値に違反したという事実が、アクティブにされたしきい値違反イベント・モニターに記録されます。しかし、さらに詳細な情報を必要とする場合には、各しきい値定義で、アクティブにされたアクティビティー・イベント・モニターに他のデータをキャプチャーすることを要求できます。例えば、個々のアクティビティーに関する情報、ステートメント・テキスト、コンパイル環境、さらには入力データ値などを要求できます。

実行の停止

しきい値の違反が発生した場合に取る一般的なアクションは、アクティビティーの実行を停止することです。この場合、サブミットしたアプリケーションに対して、しきい値に違反したことを示すエラー・コードが返されます。

実行の継続

状況によっては、アクティビティーの実行を停止するのは、行き過ぎである場合もあります。望ましい対応は、そのアクティビティーの実行は継続させ、この状態が再度発生しないようにする方法を判別するために将来分析を行うための関連データを管理者のために収集する、というものです。この状況では、サブミットしたアプリケーションにエラー・コードは返されません。アクションを続行する場合、ユーザーはしきい値に違反したという通知は受け取りません。

ワーク・アクション・セット

7 ページの『ワーク・クラス』で説明されているように、特定のタイプのアクティビティー (LOAD アクティビティー、READ アクティビティーなど) を表すワーク・クラスを定義できます。ワーク・アクションは、ワーク・クラスに適用できるアクションを提供します。ワーク・アクション・セットは、特定のスーパークラスまたはデータベース全体のいずれかのアクティビティーに適用できる 1 つ以上のワ

ーク・アクションを含んでいます。ワーク・クラスをアクティブにし、それにアクティビティーを割り当てるには、そのワーク・クラスに定義されたワーク・アクションが必要です。

ワーク・アクション・セットをデータベースに適用した場合、あるワーク・クラスに該当するアクティビティーに適用できるアクションのタイプがいくつかあります (しきい値定義、実行の回避、アクティビティー・データの収集、アクティビティーのカウントなど)。ワーク・アクションのしきい値の定義は、データベースのワーク・アクションとして最も強力なものです。例えば、SQL が 100 000 を超える行数を読み取ったり返したりしないようにする場合を考慮します。 SQL READ ステートメントを識別するワーク・アクション・セットのために単一のワーク・クラスを定義し、戻り行数が 100 000 を超えた場合に実行を停止するしきい値を持つワーク・アクションを定義できます。実行可能なアクションについて詳しくは、 97 ページの『ワーク・アクションとワーク・アクション・セットのドメイン』を参照してください。

サービス・スーパークラスのワーク・アクション・セットを定義した場合、アクティビティーに適用できるアクションのタイプには、サービス・クラスへのアクティビティーのマッピング、実行の回避、アクティビティーまたは集約アクティビティー・データの収集、アクティビティーのカウントというさまざまなものが含まれます。通常、ワーク・アクション・セットはアクティビティーをサービス・サブクラスにマップし、アクティビティーの管理の助けとなるようにそのサブクラスにしきい値を定義しています。

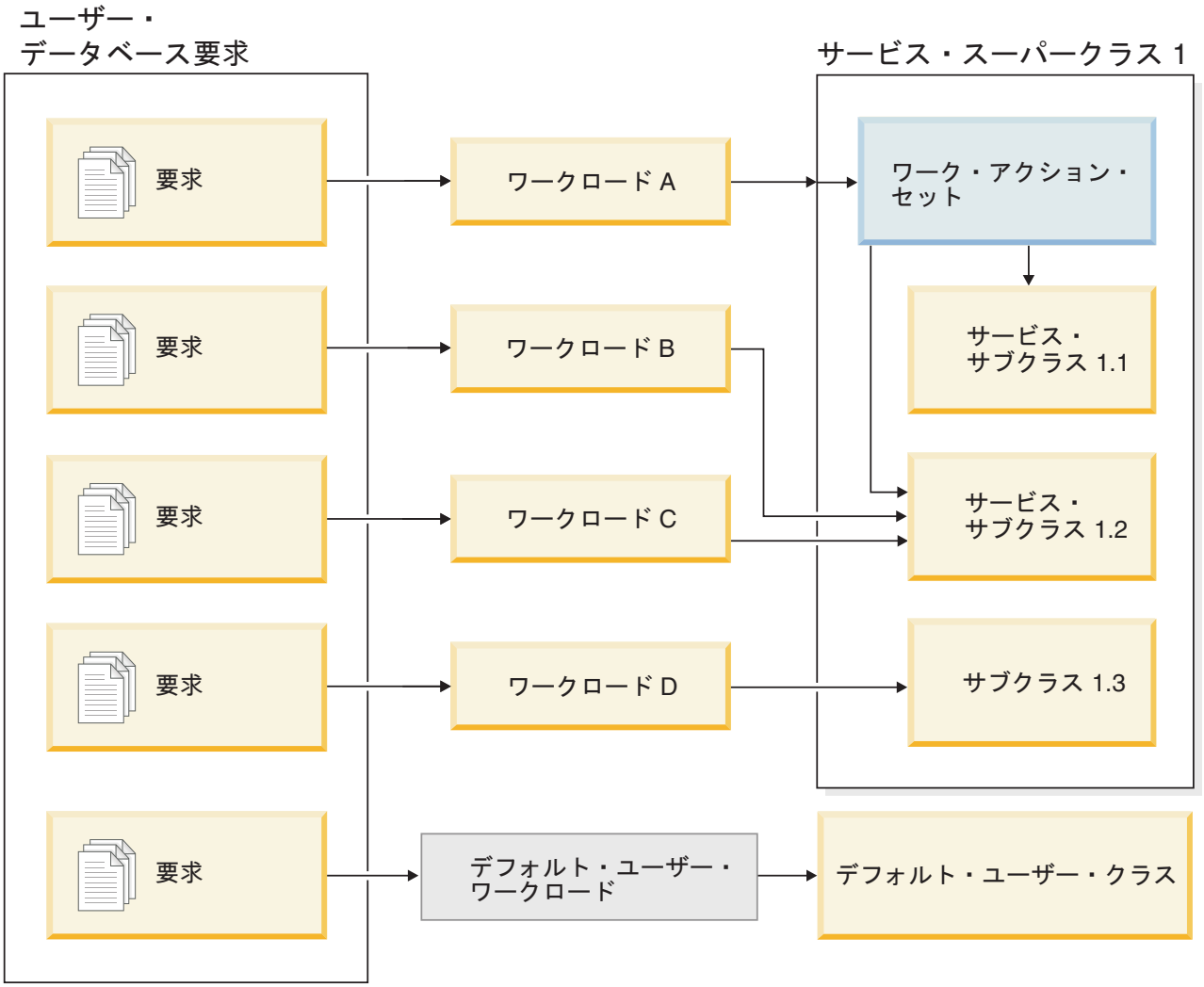


図 6. サービス・スーパークラスのためのワーク・アクション・セットのマッピング

ワークロード管理のモニター・ステージ

ワークロード管理の 3 番目のステージはモニターです。これはワークロード管理の最後のステージとして説明されていますが、モニターは継続して行われるものです。

モニターの主な目的は、システム、およびシステム上で実行される個別のワークロードの健全さと効率を確認することです。表関数を使用することより、リアルタイム運用データ (実行中のワークロード・オカレンスおよびサービス・クラスで実行中のアクティビティーのリスト、平均応答時間など) にアクセスできます。イベント・モニターを使用して、詳細なアクティビティー情報を収集したり、履歴分析のためにアクティビティーの統計を集約したりすることができます。

モニター方針を作成する時の最初のステップとして、通常、集約情報を調べます。集約情報はデータ・サーバー・アクティビティーの全体像を十分に反映していると

共に、調べる対象となるすべてのアクティビティーの情報を収集する必要がないため、安上がりです。モニターが必要な領域の範囲が明確になってきたら、さらに詳細な情報を収集できます。

リアルタイム・モニター

リアルタイム・モニターを使用して、パフォーマンス問題または問題報告書に対応してシステム上に何が起こっているかを判別することができます。

リアルタイム・モニターのデータには、使用パターンとリソース割り振りを決定したり問題のある領域を識別するために役立つ、システムでの現在のアクティビティーを示す統計が含まれます。

リアルタイム・モニターのデータにアクセスする方法は、DB2 表関数を介するものです。表関数を使用すると、SELECT ステートメントを実行できる仮想 DB2 表として DB2 データベースの内部に存在するデータ集合 (ワークロード管理統計など) にアクセスできます。これにより、データ・サーバー上の物理表の場合と同様にデータを照会してそれを分析するためのアプリケーションを作成することができます。

現在システム上で実行中の作業に関する情報のセットを返す表関数もあります。作業に関するこの情報は、さまざまなレベルにおいて得られます。

表 6. 実行中の作業に関して得られる表関数情報

情報収集の対象となるオブジェクト	**説明**
ワークロード・オカレンス	WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数は、全データベース・パーティションにおける、サービス・クラスに割り当てられたワークロード・オカレンスのリストを返します。オカレンスごとに、ワークロードをサービス・クラスに割り当てるために使用される現在の状態と接続属性に関する情報と、アクティビティー・ボリュームおよび成功率を示すアクティビティー統計に関する情報があります。
サービス・クラス・エージェント	WLM_GET_SERVICE_CLASS_AGENTS 表関数は、サービス・クラスまたはアプリケーション・ハンドルに関連付けられたデータベース・エージェントのリストを返します。返される情報は、エージェントの現在の状態、エージェントが実行中のアクション、およびそのアクションの状況も示します。
ワークロード・オカレンス・アクティビティー	WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数は、ワークロード・オカレンスに関連付けられた現在のアクティビティーのリストを返します。アクティビティーごとに、アクティビティーの現在の状態 (例えば、実行中または待機中)、アクティビティーのタイプ (例えば、LOAD、READ、DDL)、およびアクティビティーが開始した時刻に関する情報が得られます。
アクティビティーの詳細	WLM_GET_ACTIVITY_DETAILS 表関数は、個々のアクティビティーに関する詳細を返します。返される詳細の 1 つはアクティビティー・タイプです。そのタイプに応じて追加データのセットが返されます。例えば SQL アクティビティーにおいては、ステートメント・テキスト、パッケージ・データ、コスト見積もり、および戻されたり変更されたりした行に関する情報が得られます。分離レベルおよび CPU 時間に関する詳細も得られます。

一般統計情報は、以下の表で説明されているレベルを含めて、いくつかの異なるレベルでも使用できます。

表 7. ワークロード管理ソリューションで使用できる統計情報

統計の対象となるオブジェクト	**返される統計の説明**
サービス・スーパークラス	WLM_GET_SERVICE_SUPERCLASS_STATS 表関数は、サービス・スーパークラス・レベルでの全データベース・パーティションにおけるサマリー統計、つまり同時接続の最高水準点を示します (これはワークロード・アクティビティーのピークを判別する時に役立ちます)。
サービス・サブクラス	WLM_GET_SERVICE_SUBCLASS_STATS 表関数は、サービス・サブクラス・レベルでの全データベース・パーティションにおけるサマリー統計を示します (すべてのアクティビティーはサービス・サブクラスで実行されます)。統計には、完了したアクティビティー数と平均実行時間が含まれます (サービス・クラスおよびデータベース・パーティション全体の一般的なシステム・ヘルスおよびアクティビティーの分散を調べる時に役立ちます)。
ワークロード	WLM_GET_WORKLOAD_STATS 表関数は、ワークロード・レベルでの全データベース・パーティションにおけるサマリー統計を示します。これには、並行ワークロード・オカレンスの最高水準点と完了したアクティビティー数が含まれます (一般的なシステム・ヘルスのモニターの時、または問題のある領域を識別するために詳しく調べる時に役立ちます)。
ワーク・アクション・セット	WLM_GET_WORK_ACTION_SET_STATS 表関数は、ワーク・アクション・セット・レベルでの全データベース・パーティションにおけるサマリー統計、つまり対応するワーク・アクションが適用されたそれぞれのワーク・クラスのアクティビティーの数を示します (これはワーク・アクション・セットの効果性を理解し、システムで実行中のアクティビティーのタイプを理解する上で役立ちます)。
しきい値キュー	WLM_GET_QUEUE_STATS 表関数は、キューに対応するしきい値で使用される、キューの全データベース・パーティションにおけるサマリー統計を示します。統計には、キューに入れられたアクティビティーの数 (現在のキューと合計) およびキューで費やされた合計時間が含まれます (現在キューに入れられているアクティビティーを照会する場合またはしきい値が正しく定義されていることを検証する場合に役立ちます。キューイングが頻繁に発生する場合は、しきい値の制限が大きすぎることを示している可能性があり、キューイングがほとんど発生しない場合には、しきい値の制限が小さすぎるか必要ないことを示している可能性があります。)

統計は、統計が収集される期間に意味がある場合にのみ役立ちます。非常に長い時間にわたり統計を収集したり、いつでも WLM_COLLECT_STATS ストアード・プロシージャーを使用することによって、多くの古いデータが存在するようになるために傾向の変化または問題のある領域を識別することが難しくなる場合があり、あまり役立たない場合があります。そのため、いつでも統計をリセットできるようになっています。

デフォルトのワークロードおよびデフォルトのユーザー・サービス・クラスがあるため、モニター機能は DB2 データ・サーバーをインストールした時点から存在します。この機能は、ワークロード管理の識別ステージを開始または検証する上で非常に役立ちます。ワークロード、およびそれらを割り当てることができるサービス・クラスを作成するために使用できるアクティビティーのソースを容易に分離できるからです。

例: ワークロード管理表関数の使用

ワークロード管理のリアルタイム・モニターで大量のデータが得られます。このトピックの例では、その情報を使えるようにする方法を示します。

ここでは、デフォルトのワークロードおよびサービス・クラスのみが使用されている状態を考えます。この例を使うと、データ・サーバーで何が実行されているかを正確に把握するためにどのように表関数を使用できるかを理解できます。以下のステップを実行してください。

1. サービス・スーパークラス統計表関数を使用してすべてのサービス・スーパークラスを表示します。 DB2 9.5 をインストール、またはそのバージョンにマイグレーションした後、3 つのデフォルトのスーパークラスが定義されます。つまり、1 つは保守アクティビティー用、1 つはシステム・アクティビティー用、もう 1 つはユーザー・アクティビティー用です。ここで使用するサービス・クラスは SYSDEFAULTUSERCLASS です。
2. サービス・サブクラス統計表関数を使用して、SYSDEFAULTUSERCLASS スーパークラスのすべてのサービス・サブクラスの統計を表示します。それぞれのサービス・サブクラスについては、現在処理中の要求の量、実行が完了したアクティビティーの数、および全データベース・パーティションにおけるアクティビティーの全体的な分散 (分散が不均一である場合、問題がある可能性があります) が示されます。オプションで、アクティビティーの平均存続時間、アクティビティーがキューで費やす平均時間などを含む追加の統計を取得することができます。ALTER SERVICE CLASS ステートメント上の COLLECT AGGREGATE ACTIVITY DATA キーワードを指定して、集約アクティビティーの統計の収集を有効にすることにより、サービス・サブクラスのオプションの統計を取得できます。
3. 特定のサービス・サブクラスにおいては、ワークロード・オカレンス情報表関数を使用して、サービス・サブクラスにマップされるワークロードのオカレンスをリストします。表関数は、すべての接続属性を表示します。これを使用してアクティビティーのソースを識別できます。この情報は、将来カスタム・ワークロード定義を決定する上で非常に役立つ場合があります。例えば、ここでリストされる特定のワークロード・オカレンスには、完了したアクティビティーのカウンターで示されるようにアプリケーションからの大量の作業があるかもしれません。
   a. そのアプリケーションでは、ワークロード・オカレンス・アクティビティー情報表関数を使用して、アプリケーションの接続により作成された、全データベース・パーティションにおける現在のアクティビティーを表示します。この情報は、データ・サーバー上で問題の原因となっている可能性があるアクティビティーの識別など、多くの目的に使用できます。
   b. アクティビティーごとに、アクティビティーの詳細表関数を使用してさらに詳細な情報を取り出します。このデータは、大量の行を返している SQL ステートメントがあること、長時間アイドル状態であるアクティビティーがあ

ること、または極めて大きな見積コストを持つ照会が実行中であることを示している可能性があります。このような状況では、将来の損害を与える可能性のある動作を識別して防ぐために何らかのしきい値を定義することには意味があると考えられます。

履歴モニター

DB2 ワークロード・マネージャーは、表関数に加えて、イベント・モニターを使用して将来または履歴分析のために活用できる可能性がある情報をキャプチャーします。

ワークロード管理の構成で使用できる 3 つのイベント・モニターがあります。それぞれのイベント・モニターは、それぞれ異なる目的で使用されます。

アクティビティー・イベント・モニター

このモニターは、サービス・クラス、ワークロード、またはワーク・クラスにおける個々のアクティビティーに関する情報またはしきい値に違反したアクティビティーに関する情報をキャプチャーします。アクティビティーごとにキャプチャーされるデータの量は構成可能で、ディスク・スペースの容量およびモニター・データを保持しなければならない時間の長さを決定する際に考慮する必要があります。アクティビティー・データの一般的な使用法としては、db2advis などのツールへの入力データとして、または一連の照会における表、列、および索引の使用量の判別に役立つアクセス・プランを (Explain ユーティリティーから) 使用するために活用するというものです。

しきい値違反イベント・モニター

このモニターは、しきい値に違反した場合に情報をキャプチャーします。これは、どのしきい値に違反したのか、違反の原因となったアクティビティー、およびそれが発生した時に取られたアクションを知らせます。

統計イベント・モニター

このモニターは、詳細なアクティビティー情報をキャプチャーする代わりに、集約データ (完了したアクティビティーの数、平均実行時間など) を収集することによりオーバーヘッドを少なくします。集約データには、存続時間、キュー時間、実行時間および見積コストを含むさまざまなアクティビティー測定のヒストグラムが含まれます。ヒストグラムを使用して値の分布を理解し、異常値を識別し、そして平均および標準偏差のような追加統計を計算することができます。例えば、ヒストグラムはユーザーの体感する応答時間の変化を理解するのに役立ちます。変動性が高い場合、平均の存続時間そのものは、ユーザーの体感する応答時間を反映するものとはなりません。

以下の図は、ワークロード情報にアクセスするために使用できるさまざまなモニター・オプションを示しています。それには、リアルタイム統計とアクティビティーの詳細にアクセスする表関数、および効率的な集約または詳細な個々のアクティビティーとしてキャプチャーされる履歴情報が含まれます。

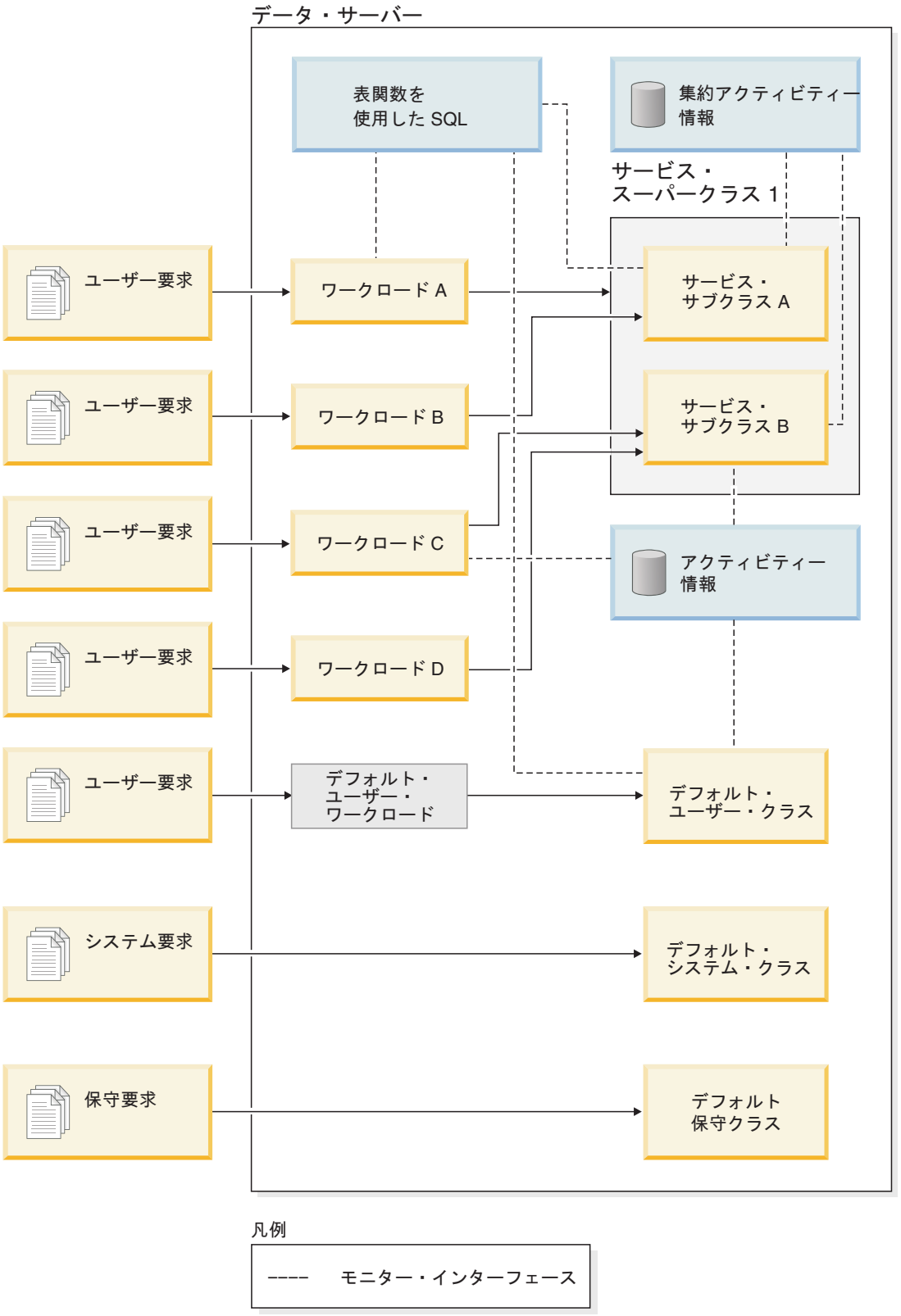


図 7. モニターを使用したワークロード管理

アクティビティー

アクティビティーとは、その存続時間中にデータベース・リソースを消費する個々の作業のことをいいます。存続時間は、1 つのデータベース要求の間の場合もありますし、複数のデータベース要求にまたがる場合もあります。CALL ステートメントまたはカーソルはアクティビティーの例です。OPEN および FETCH 要求は、カーソル・アクティビティーの存続期間中に発生するデータベース要求の例です。

データ・サーバーは、ワークロード管理構成に関与する以下のアクティビティーを認識します。

- すべての DML ステートメント
- すべての DDL ステートメント
- CALL ステートメント
- ロード・ユーティリティー

上に挙げた前のアクティビティーのいずれにも分類されない作業も、作業がサブミットされる接続の属性に基づいてワークロード (およびワークロード定義で指定された対応するサービス・クラス) に割り当てられます。ただし、この作業にしきい値またはワーク・アクションは適用されません。

アクティビティーのコンテキストでは、ネストとは、あるアクティビティーが別のアクティビティーを呼び出す状態のことを指しています。例えば、ストアード・プロシージャーは 1 つのアクティビティーです。ストアード・プロシージャーに DML アクティビティーが含まれている場合、そのアクティビティーはネストされたアクティビティーということになります。ストアード・プロシージャーが別のストアード・プロシージャーを呼び出す場合、2 番目のストアード・プロシージャーがネストされたアクティビティーになります。

ワークロード管理のサンプル・アプリケーション

包括的なワークロード管理がバージョン 9.5 に統合され、結果としてアクティビティー、リソース、およびパフォーマンスのより細かな制御が可能になるとともに、システムの実行状況の洞察を深めることができるようになりました。現在、ワークロード管理のサンプル・アプリケーションは developerWorks® で入手できます。

ワークロード管理のサンプル・アプリケーションは、ワークロード管理フィーチャーを使用して以下の目標を達成する方法を説明します。

照会の暴走からシステムを保護する

照会の暴走は多くのコストを費やし、パフォーマンス低下の原因となります。ワークロード管理のサンプル・アプリケーションは暴走する可能性のある照会を識別し、そのような照会が指定されたしきい値に違反した後、その実行を停止させます。

個別のアプリケーションによるリソースの同時使用量を制限する

サンプル・アプリケーションは、ワークロード管理フィーチャーを使用して、大量の並行作業をサブミットするアプリケーションが他のアプリケーションのパフォーマンスにマイナスの影響を与えないようにする方法を示します。

特定の応答時間を達成する

ワークロード管理フィーチャーによって、他のどんなアクティビティーがシステム上で同時に実行されているかにかかわりなく「アプリケーション Y からのトランザクション X は 90% のケースにおいて 1 秒以内で完了する」といった形式の特定の回答時間の目標を達成することができます。サンプル・アプリケーションは回答時間の目標を達成する方法を示します。

短い照会の一貫性のある応答時間

一般的に応答時間が 1 秒未満の照会は、他のどんなワークロードがシステム上で実行されているかにかかわりなく、応答時間が比較的に一貫性のあるものとなります。サンプル・アプリケーションは照会実行時間のヒストグラムを使用して一貫性をモニターします。

ピーク要求の期間中にシステムを保護する

ワークロード管理ポリシー・フィーチャーは、いったんシステムに十分な負荷がかかると作業をキューイングして、ピーク要求のバースト中にシステムをキャパシティーの過負荷から保護します。

抽出、トランスフォーム、およびロード (ETL) のバッチ処理とユーザー照会を並行して使用可能にする

ワークロード管理フィーチャーを使用すれば、(例えば表へのデータのロードなどの) ETL ジョブを実行しながら、同時に照会を実行するユーザーのパフォーマンスへの影響を制御できます。

サンプル・アプリケーションを入手するには、developerWorks のワークロード管理のサンプルを参照してください。

第 2 部 識別および管理

© Copyright IBM Corp. 2007, 2008

第 2 章 サービス・クラス

すべてのデータベース要求は、サービス・クラスで実行されます。サービス・クラスは、すべてのデータベース要求へのリソース割り当ての 1 次点になります。さらに、サービス・クラスは、データベース内のデータベース・アクティビティーのセットをモニターおよび制御するためにも使用されます。

すべての DB2 サービス・クラスは、サービス・スーパークラスまたはサービス・スーパークラスのサービス・サブクラスのいずれかです。各データベースには複数のサービス・スーパークラスを含めることができ、各サービス・スーパークラスには複数のサービス・サブクラスを含めることができます。サービス・サブクラスは、1 つのサービス・スーパークラスにのみ属することができます。スーパークラスのリソースは、その中のすべてのサブクラスによって共有されます。

組織のパフォーマンス目標を満たすことができるよう、サービス・クラスをセットアップし、データベース内のアクティビティーを編成することができます。以下の図に示すとおり、サービス・クラスなしでは、要求を編成して認識可能な論理グループにすることはできません。

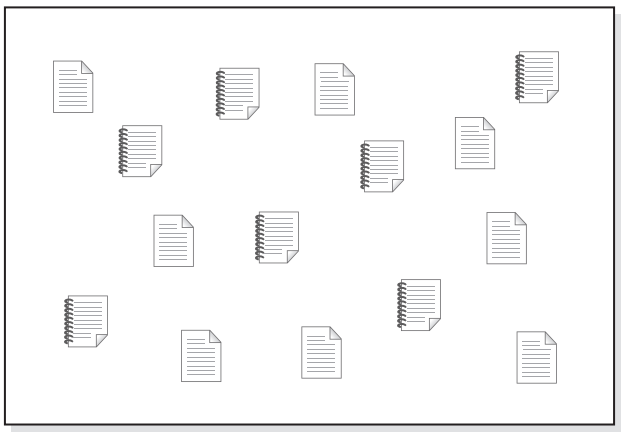


図 8. 未編成の作業

さまざまなサービス・スーパークラスを作成し、さまざまなタイプの作業用の実行環境を提供してから、該当する要求をサービス・スーパークラスに割り当てることができます。 2 つの別個の基幹業務 (金融および在庫) のアプリケーションを所有していると想定します。各基幹業務には、組織に対する責任を実行するための独自のアプリケーションがあるかもしれません。要求は、ワークロード管理目標にとって意味を成すカテゴリーに編成することができます。以下の図では、異なるサービス・スーパークラスが異なる基幹業務に割り当てられています。

© IBM Corporation 2007, 2008

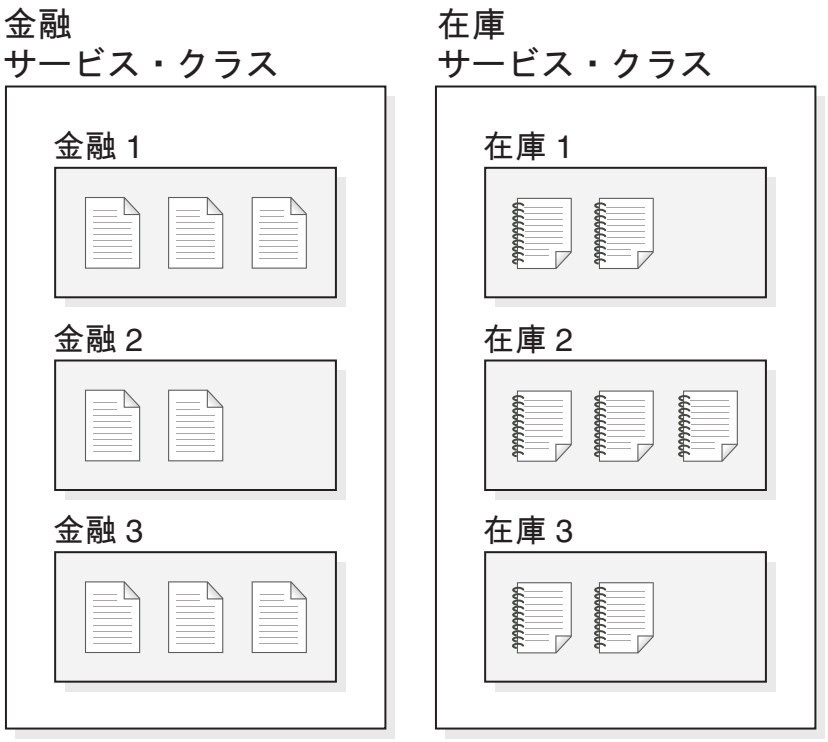


図 9. サービス・クラスによって編成される作業

前の図では、両方のサービス・スーパークラスにあるアクティビティーはさらに分割されています。サービス・クラスは、サービス・スーパークラスとその下のサービス・サブクラスという 2 層の階層を提供します。この階層により、実行環境をより複雑に分割して、実際のモデルをより良くエミュレートすることができます。特に指定がない限り、サービス・サブクラスはサービス・スーパークラスからの特性を継承します。サービス・サブクラスは、サービス・スーパークラスの作業をさらに分割するために使用します。

すべての作業は実際にはサービス・サブクラスで実行され、サービス・スーパークラスの作成時にデフォルトのサービス・サブクラスが自動的に作成されます。そのデフォルトのサブクラスは、サービス・スーパークラスにマップされる (そして残される) すべての作業で使用されるものです。サービス・スーパークラスは、そのサービス・サブクラスすべてに対する共通のバックグラウンドとして機能します。ヒストグラムおよび COLLECT ACTIVITY DATA オプションを除き、特に指定されていない限り、サービス・サブクラスはそのサービス・スーパークラスの属性を継承します。

別の作業には別の優先順位があるため、サービス・クラスを使用する場合、そのサービス・クラスの複数の特性を制御することができます。以下に例を示します。

- サービス・クラスでの作業に使用する、入出力ページのプリフェッチャーの優先順位を設定できます。
- サービス・クラスのすべてのエージェント用のエージェント CPU の優先順位を設定できます。

- さまざまなタイプのしきい値を使用して、サービス・クラスで実行中の作業に対して制約を適用することによって、サービス・クラスでのリソースの使用を制御することができます。

ワークロードを使用して、作業をサービス・スーパークラスに割り当てます。さらに、ワークロード定義またはワーク・アクションを使用して、作業をサービス・スーパークラスのサービス・サブクラスに割り当てることもできます。

サービス・クラスは CREATE SERVICE CLASS ステートメントを使用して作成できます。サービス・クラスは ALTER SERVICE CLASS ステートメントを使用して変更できます。サービス・クラスは DROP SERVICE CLASS ステートメントを使用してドロップできます。

SYSCAT.SERVICECLASSES カタログ・ビューを照会して、サービス・クラスを表示できます。

デフォルトのサービス・スーパークラスおよびサブクラス

DB2 バージョン 9.5 以降をインストールするかそれにマイグレーションすると、それぞれの新規データベースまたはマイグレーション済みデータベースには、事前定義された 3 つのデフォルトのサービス・スーパークラスが含まれることになります。それらは、デフォルトのユーザー・クラス、デフォルトの保守クラス、およびデフォルトのシステム・クラスです。

デフォルトのサービス・スーパークラスはいずれも、使用不可にしたりドロップしたりすることはできません。

デフォルトのサービス・スーパークラスはすべて、1 つのデフォルト・サービス・サブクラスとともに作成されます。デフォルトのサービス・スーパークラスの追加のサービス・サブクラスを作成することはできません。デフォルトのサービス・サブクラスは、次のように常に SYSDEFAULTSUBCLASS という名前で作成されます。

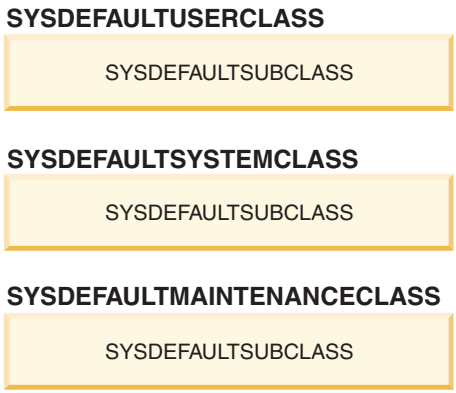


図 10. 2 層のサービス・クラス階層

デフォルトのサービス・スーパークラスへの接続によって発行される作業はすべて、そのサービス・スーパークラスのデフォルトのサービス・サブクラスで処理されます。

デフォルトのサービス・スーパークラスおよびそのデフォルトのサービス・サブクラスがドロップされるのは、データベースがドロップされる場合のみです。 DROP SERVICE CLASS ステートメントを使用してこれらをドロップすることはできません。

デフォルトのユーザー・サービス・スーパークラス (SYSDEFAULTUSERCLASS)
DB2 バージョン 9.5 をインストールまたはそれにマイグレーションした後に、デフォルトでは、すべてのアクティビティーが SYSDEFAULTUSERCLASS で実行されます。

デフォルトの保守サービス・スーパークラス (SYSDEFAULTMAINTENANCECLASS)
デフォルトの保守サービス・スーパークラスは、データベース保守および管理タスクを実行する内部 DB2 接続をトラッキングします。 DB2 非同期バックグラウンド・プロセス (ABP) エージェントからの接続は、このサービス・スーパークラスにマップされます。 ABP エージェントは、データベース保守タスクを実行する内部エージェントです。非同期索引のクリーンアップ (AIC) は ABP 主導タスクの一例です。ABP エージェントは、データ・サーバー上でユーザー接続の数が増加すると、リソースの使用量およびサブエージェントの数を自動的に削減します。ユーザー接続によって発行されるユーティリティーは、通常のサービス・クラスを使用してマップされます。サービス・クラスのしきい値を SYSDEFAULTMAINTENANCECLASS にインプリメントすることはできません。

デフォルトの保守サービス・スーパークラスによってトラッキングされる内部接続には、以下のものが含まれます。

- ABP 接続 (AIC を含む)
- ヘルス・モニターによって開始されたバックアップ
- ヘルス・モニターによって開始された RUNSTATS
- ヘルス・モニターによって開始された REORG

デフォルトのシステム・サービス・スーパークラス (SYSDEFAULTSYSTEMCLASS)
デフォルトのシステム・サービス・スーパークラスは、システム・レベルのタスクを実行する内部 DB2 接続およびスレッドをトラッキングします。このサービス・スーパークラスに対してサービス・サブクラスを定義したり、ワークロードまたはワーク・アクションを関連付けたりすることはできません。さらに、サービス・クラスのしきい値を SYSDEFAULTSYSTEMCLASS にインプリメントすることはできません。デフォルトのシステム・サービス・スーパークラスによってトラッキングされる DB2 スレッドおよび接続には、以下のものが含まれます。

- ABP デーモン
- Query Patroller (QP) 接続
- セルフチューニング・メモリー・マネージャー (STMM)
- プリフェッチャー・エンジン・ディスパッチ可能単位 (EDU) (db2pfchr)
- ページ・クリーナー EDU (db2pclnr)
- ログ読み取りプログラム EDU (db2loggr)
- ログ書き込みプログラム EDU (db2loggw)

- ログ・ファイル読み取りプログラム EDU (db2lfr)
- デッドロック検出機能 EDU (db2dlock)
- イベント・モニター (db2evm)
- システム・レベルのタスクを実行する接続

Query Patroller 接続は、QP の開始時に QP コントローラー (QP のサーバー・コンポーネント) によって発行される DB2 データ・サーバーへの内部的な接続です。この接続は QP の始動時に確立され、QP が正常に開始した後は、デフォルトのシステム・サービス・スーパークラスに接続がマップされます。 QP の始動中に、通常のワークロード・マッピング処理の一環として、接続が別のサービス・クラスに一時的にマップされる場合があります。この期間中に、接続は、一時的なマップ先のサービス・クラスのすべてのコントロールおよびしきい値に従います。

アクティビティーからサービス・クラスへのマッピング

すべてのデータベース接続は、最初の作業単位の開始時にワークロードに割り当てられます。ワークロード・オカレンスが開始されると、ワークロード定義で指定されたサービス・クラス名に基づいて、そのワークロード・オカレンスの下で実行されているすべてのアクティビティーがサービス・クラスにマップされます。ワークロード・オカレンスがサービス・スーパークラスに割り当てられると、ワーク・アクション・セットがそのサービス・スーパークラスに定義されている場合、そのワークロード・オカレンスにサブミットされたすべての作業を、(ワーク・クラスに適用された) ワーク・アクションによってそのサービス・スーパークラス内のユーザー定義のサービス・サブクラス (つまり、デフォルトのサービス・サブクラスではない) に再割り当てすることができます。

接続がワークロード定義に定義された基準を満たしている場合、データ・サーバーはそのワークロード定義に接続を割り当てます。例えば、アプリケーション A からの接続はすべてワークロード定義 Alpha に属するように、アプリケーション B からの接続はすべてワークロード定義 Beta に属するように、ワークロード管理インプリメンテーションをセットアップすることができます。

CREATE WORKLOAD ステートメントの SERVICE CLASS キーワードを指定することによって、ワークロードを使用してアクティビティーを接続からサービス・スーパークラスにマップできます。アクティビティーに当てはまるワーク・クラスまたはワーク・アクションがない場合、アクティビティーはサービス・スーパークラスのデフォルトのサービス・サブクラスで実行されます。

さらに、CREATE WORKLOAD ステートメントの SERVICE CLASS キーワードで UNDER キーワードを指定することによって、ワークロードを使用してアクティビティーを接続からサービス・スーパークラスのサービス・サブクラスにマップすることもできます。この場合、接続は依然としてサービス・スーパークラスに属していますが、その接続から発行されたすべてのアクティビティーは、ワークロード定義で指定されたサービス・サブクラスに自動的にマップされます。

注: 接続用のサービス・スーパークラスのマッピングを行うのは、コーディネーター・エージェントのみです。コーディネーター・エージェントがサブエージェントを作成する場合、サブエージェントはコーディネーター・エージェントのスーパークラス・マッピングを継承します。

以下の図では、接続、ワークロード、およびサービス・スーパークラスの間の関係を示しています。ワークロード A の定義を満たす接続は、サービス・スーパークラス 1 にマップされます。ワークロード B または C の定義を満たす接続はサービス・スーパークラス 2 にマップされます。ワークロード D の定義を満たす接続は SYSDEFAULTUSERCLASS サービス・スーパークラスにマップされます。

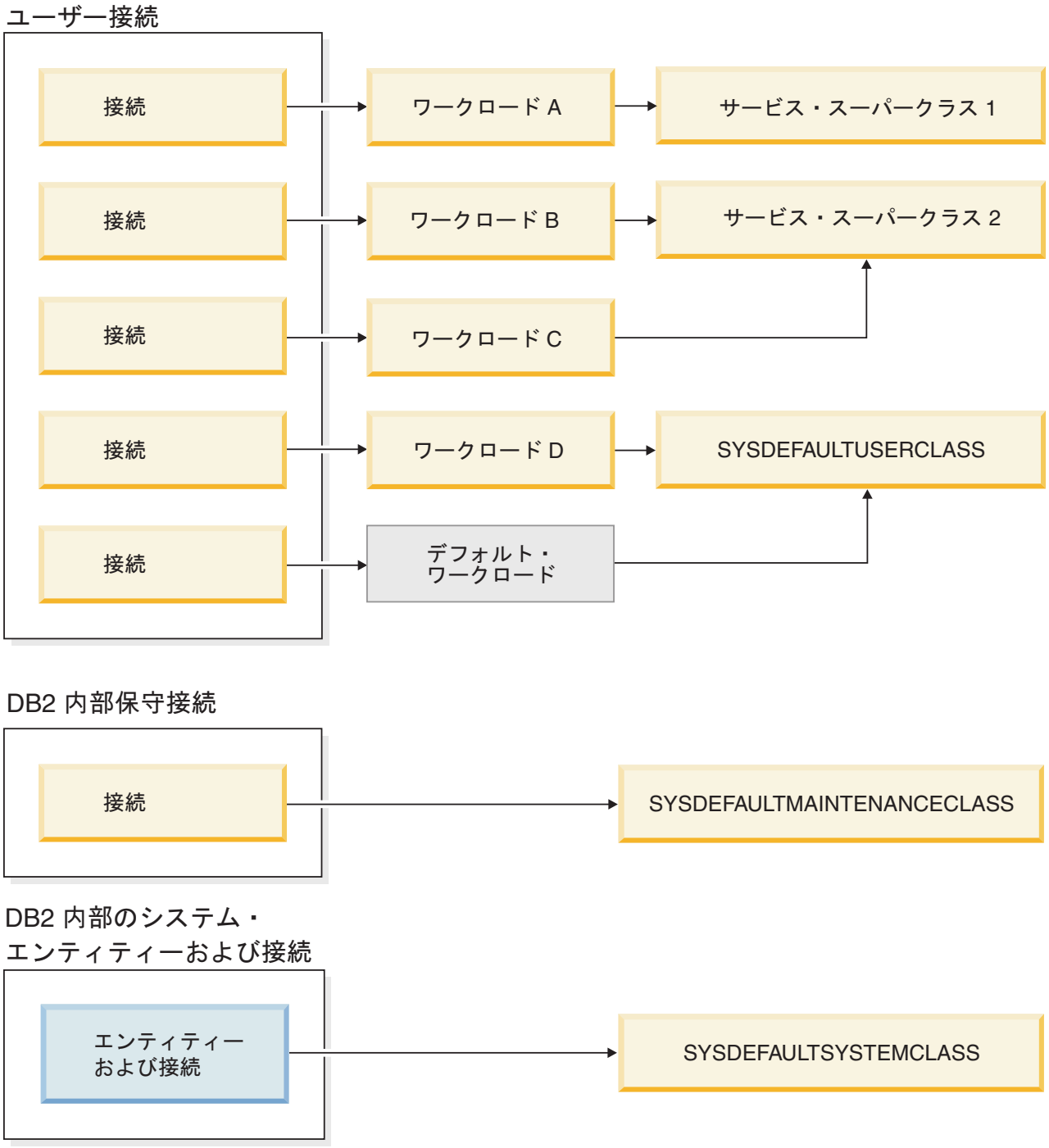


図 11. サービス・スーパークラスへのデータベース接続のマッピング

ワークロード管理構成がより複雑な場合、アクティビティー・タイプまたは他のアクティビティー属性に基づいてアクティビティーを別々に処理することも可能です。例えば、以下のアクションのいずれかを実行することができます。

- DML を DDL とは異なるサービス・サブクラスに配置する。
- 見積コストが 100 timeron 未満のすべての読み取りタイプ照会を、その他すべての読み取りタイプ照会とは異なるサービス・サブクラスに配置する。

より複雑な構成では、アクティビティーを接続からサービス・スーパークラスにマップするよう、ワークロードをセットアップすることができます。続いて、ワーク・アクション (サービス・スーパークラスに適用されるワーク・アクション・セットに入っている) を使用して、アクティビティーをそのタイプまたは属性に基づいて、サービス・スーパークラスの特定のサービス・サブクラスに再マップすることができます。

特に、MAP ACTIVITY ワーク・アクションが入っているワーク・アクション・セットをサービス・スーパークラスに適用することができるかもしれません。サービス・スーパークラスにマップされ、かつ MAP ACTIVITY ワーク・アクションに関連付けられているワーク・クラスと一致するすべてのアクティビティーは、ワーク・アクションによって指定されたサービス・サブクラスにマップされます。

ワークロードがアクティビティーをサービス・サブクラスにマップする場合、そのアクティビティーは、サービス・スーパークラスに適用されるワーク・アクション・セット内のワーク・アクションによって影響を受けることはありません。

- ワークロードによって、アクティビティーをサービス・スーパークラスの 1 つのサービス・サブクラスにマップできる。
- 同じサービス・スーパークラス内の別のサービス・サブクラスへとアクティビティーをマップするワーク・アクションも、このアクティビティーに適用される。

ワークロードまたはワーク・アクションによってアクティビティーがサービス・サブクラスにマップされない場合、そのアクティビティーは、そのアクティビティーのサービス・スーパークラスのデフォルトのサブクラス (SYSDEFAULTSUBCLASS) にマップされます。

データベース・アクティビティーがそれぞれのサービス・スーパークラスおよびサービス・サブクラスにマップされている場合、特定のサービス・クラス内のすべてのアクティビティーを制御することができます。統計は、そのサービス・クラスのデータベース・アクティビティーをモニターするために使用できるサービス・クラス・レベルで使用可能です。

以下の図では、ワークロードを介してサービス・スーパークラスまたはサービス・サブクラスにマップされているデータベースへの接続を示します。ワーク・アクションを使用してアクティビティーをサービス・サブクラスにマップする方法については、90 ページの『ワーク・アクションとワーク・アクション・セット』を参照してください。

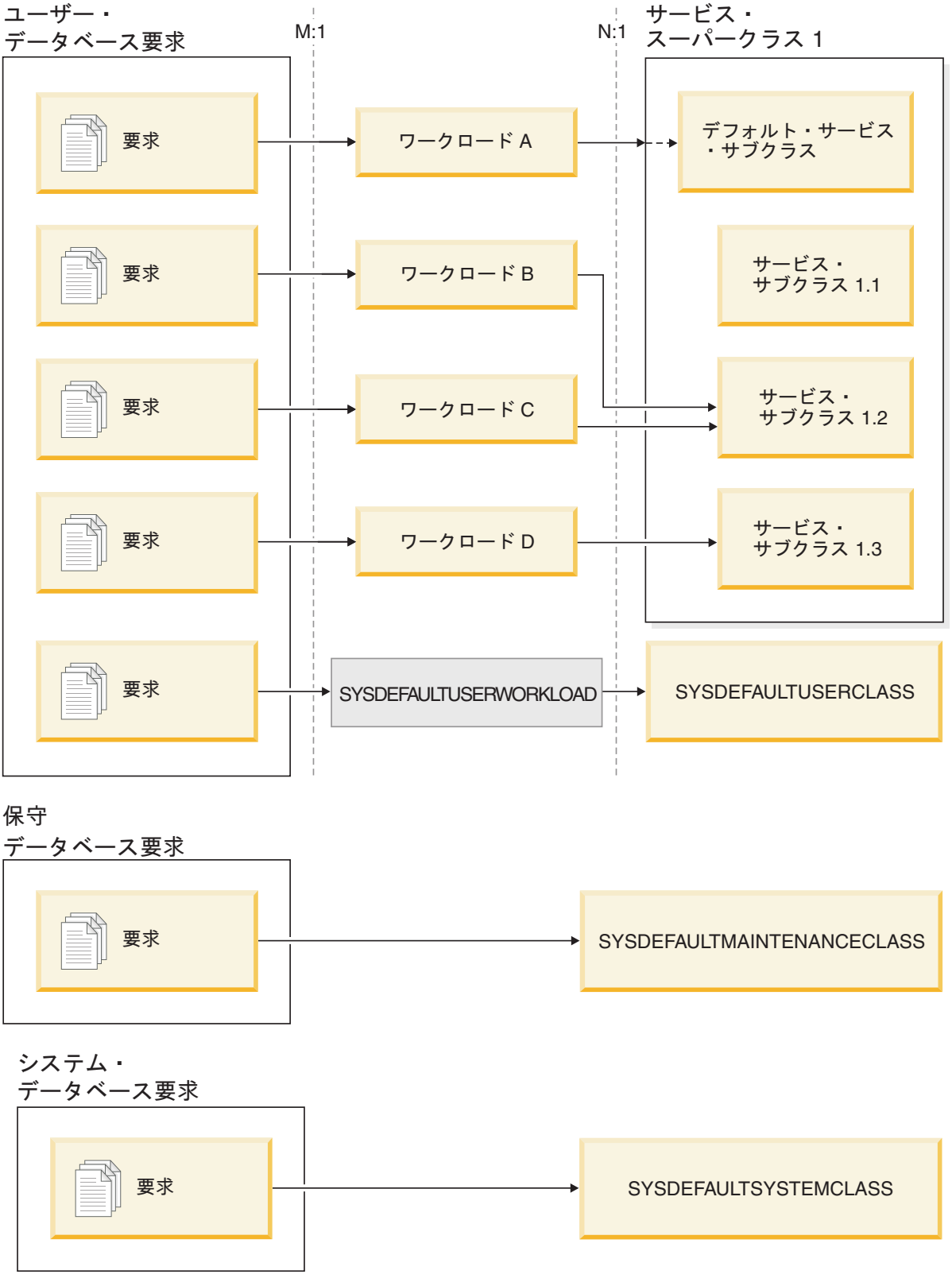


図 12. サービス・スーパークラスにマップされているデータベース接続

ユーザー定義のワークロード定義にマップしない接続は、デフォルトのユーザー・ワークロード定義 SYSDEFAULTUSERWORKLOAD にマップされます。デフォルト

で、デフォルトのワークロード定義 (SYSDEFAULTUSERWORKLOAD) からの接続は、ユーザー要求に対するデフォルトのサービス・スーパークラスである SYSDEFAULTUSERCLASS サービス・スーパークラスにマップされます。SYSDEFAULTUSERWORKLOAD ワークロードは、別のサービス・クラスにマップするように変更することができます。内部 DB2 保守接続は、保守要求のデフォルトのサービス・スーパークラスである SYSDEFAULTMAINTENANCECLASS にマップされます。内部システム・エンティティーおよび接続は、システム・レベルのタスクを実行する内部 DB2 接続およびスレッドのデフォルトのサービス・スーパークラスである SYSDEFAULTSYSTEMCLASS にマップされます。

CPU 優先順位および DB2 サービス・クラス

DB2 サービス・クラスを使用して、各サービス・クラスを相対的なエージェントの優先順位と関連付けることができます。この優先順位は、サービス・クラスで機能するすべてのエージェントに対して設定され、他のすべての DB2 エージェントのエージェント優先順位に対する相対的なものです。サービス・クラスに対してエージェントの優先順位の値を指定しなかった場合、そのサービス・クラスのすべてのエージェントの優先順位は、他のすべての DB2 エージェントと同じになります。

DB2 サービス・クラスのエージェント優先順位を設定すると、新規作業が入れられるそのサービス・クラスで実行中のエージェントの優先順位のみが調整されます。サービス・クラスで実行するエージェント以外のスレッドは、サービス・クラスに指定されたエージェントの優先順位の値を使用しません。

DB2 サービス・クラスと AIX ワークロード・マネージャーを統合する場合、AIX ワークロード・マネージャーを使用して、オペレーティング・システムのクラスで使用する CPU の優先順位を指定してから、DB2 サービス・クラスの OUTBOUND CORRELATOR 値を使用して DB2 サービス・クラスにこの値を継承させることができます。この場合、オペレーティング・システム・ワークロード・マネージャーを使用して指定する CPU の優先順位は、DB2 サービス・クラスで実行されるエージェントの優先順位を制御し、すべてのサービス・クラス・エージェントの優先順位設定は無視されます。

agentpri データベース・マネージャー構成パラメーターは、DB2 インスタンス内のすべてのエージェントの絶対的な CPU 優先順位を固定値に設定します。エージェントに対して絶対優先順位が設定されている場合、その相対優先順位を変更することはできません。このため、**agentpri** は、サービス・クラスのエージェント優先順位あるいは AIX ワークロード・マネージャーとの統合との互換性はありません。**agentpri** が非デフォルト値に設定されている場合、サービス・クラスのエージェント優先順位および AIX ワークロード・マネージャーはエージェントの優先順位に影響しません。ワークロード管理構成では、この非推奨構成パラメーターを使用しないでください。

AIX で、AGENT PRIORITY を使用してサービス・クラスのエージェントにより高い相対優先順位を設定するためには、インスタンスの所有者に CAP_NUMA_ATTACH 機能および CAP_PROPAGATE 機能が必要であることに注意してください。これらの機能を付与するには、root としてログオンし、次のコマンドを実行します。

```
chuser capabilities=CAP_NUMA_ATTACH,CAP_PROPAGATE
```

サービス・クラスのプリフェッチ優先順位

プリフェッチャーは、ディスクからデータを検索し、このデータをバッファー・プールに保管して、エージェントがデータに即時にアクセスできるようにします。DB2 ワークロード管理では、各サービス・スーパークラスおよびサブクラスが異なるプリフェッチ優先順位を持つように割り当てることができます。

エージェントは、先読み要求をデータベース・プリフェッチ・キューに送信します。プリフェッチャーは、これらの先読み要求をキューから取り、そのデータをバッファー・プールに取り出します。特定のデータを必要とする場合、エージェントは最初にバッファー・プールを検査して、データが使用可能かどうかを確認します。使用可能でない場合、エージェントはディスクからデータを取り出します。プリフェッチャーは、長い時間がかかるディスク入出力操作を実行し、エージェントを解放して処理を並行して実行できるようにします。

サービス・クラスにルーティングされるすべての接続において、プリフェッチ要求はサービス・クラスに割り当てられたプリフェッチ優先順位に従って処理されます。各サービス・クラスは、高、中、または低の 3 つのプリフェッチ優先順位のいずれかと関連付けることができます。 CREATE または ALTER SERVICE CLASS ステートメントのいずれかで、PREFETCH PRIORITY キーワードを使用して、サービス・クラスのプリフェッチ優先順位を指定します。

サービス・スーパークラスに DEFAULT を指定すると、サービス・スーパークラスのプリフェッチ優先順位は中に設定されます。サービス・スーパークラスのすべてのサービス・サブクラスで異なるプリフェッチ優先順位を指定できますが、サービス・サブクラスに対してデフォルトのプリフェッチ優先順位を使用する場合、サービス・サブクラスはそのサービス・スーパークラスからプリフェッチ優先順位の設定を継承します。

高優先順位プリフェッチ要求は、中優先順位プリフェッチ要求より前に処理され、中優先順位プリフェッチ要求は、低優先順位プリフェッチ要求より前に処理されます。

サービス・クラスにおける接続およびアクティビティーの状態

サービス・クラスは、サービス・クラスごとに接続統計を収集します。あるサービス・クラスにどの接続およびアクティビティーがあるのか、またその接続またはアクティビティーの状態を表示することができます。

接続の状態

サービス・クラスにおける接続の状態として、以下が考えられます。

接続済み
この接続はデータベースに正常に接続されていますが、まだそのワークロードおよびサービス・スーパークラスには関連付けられていません。

マップ済み
この接続はワークロードにマップされていて、サービス・スーパークラスに加わりました。この接続はアクティビティーをサブミットして実行することができるようになりました。

過渡 この接続は、接続しきい値に達したサービス・クラスに加わろうとしています。この接続はサービス・クラスに加わるためのキューに入れられています。サービス・クラスが接続しきい値に違反していなければ、その接続はサービス・クラスに加わります。過渡状態の接続は、アクティビティーをサブミットして実行することができません。

終了中 この接続は、クライアントからの接続リセットを受け取ったか、強制終了またはエラー状態のために終了中です。

アクティビティーの状態

サービス・クラスにおけるアクティビティーの状態として、以下が考えられます。

初期化中
このアクティビティーは作成され、実行のために準備中です。

待機中 このアクティビティーは、データベース・レベルまたはサービス・クラス・レベルの並行性しきい値のために実行できません。このアクティビティーは実行が許可されるまでキューに入れられます。

注: AIX オペレーティング・システムでは、キューに入れられたアクティビティーが SQL4297N を受け取った場合、DB2 クライアントとデータ・サーバーに次の APAR がインストールされていることを確認します。

- AIX 5.3 の場合は IY89429
- AIX 5.2 の場合は IY89387

実行中 このアクティビティーは実行中です。

QP 待機中
このアクティビティーは Query Patroller によりキューに入れられています。

終了中 このアクティビティーは終了します。

サービス・クラスによってトラッキングされないシステム・レベルのエンティティー

サービス・クラスは、データベース・レベルでオブジェクトをモニターおよび制御するために使用されます。ただし、すべての DB2 エンティティーがデータベースで直接機能するわけではありません。

サービス・クラスはデータベース内で機能し、データベースのカタログ表に保管されるため、データベースで機能しないエンティティーをサービス・クラスによってトラッキングすることはできません。システム・コントローラーおよびヘルス・モニター・デーモンなどのインスタンス・レベルのエンティティーは、インスタンス・レベルで機能し、他のデータベースと直接関連付けられることはありません。インスタンス・アタッチメントおよびゲートウェイ接続を実行するエージェントも、サービス・クラスによってトラッキングされません。インスタンス・アタッチメント・エージェントおよびゲートウェイ・エージェントはデータベース内では機能しないため、これらはサービス・クラスによってトラッキングされません。

以下のリストは、データベース内で機能しないエンティティーの部分リストであり、サービス・クラスによってトラッキングされません。

- DB2 システム・コントローラー (db2sysc)
- IPC リスナー (db2ipccm)
- TCP リスナー (db2tcpcm)
- FCM デーモン (db2fcms、db2fcmr)
- DB2 再同期エージェント (db2resync)
- アイドル・エージェント (データベースとの関連がないエージェント)
- インスタンス・アタッチメント・エージェント
- ゲートウェイ・エージェント
- その他すべてのインスタンス・レベルの EDU

サービス・クラスの作業

サービス・クラスの作成

DDL ステートメント CREATE SERVICE CLASS を使用して、サービス・スーパークラスとその下のサービス・サブクラスを作成します。

サービス・クラスを作成するためには、DBADM、SYSADM、または SYSCNTRL 権限が必要です。

その他の前提条件について、以下のトピックも参照してください。

- 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』
- 命名規則

サービス・クラスを作成するには、次のようにします。

1. CREATE SERVICE CLASS ステートメントに、以下に挙げるサービス・クラスのプロパティーを 1 つ以上指定します。
   - サービス・クラスの名前を指定。

     注: サービス・クラスの名前は、一度設定すると変更することができません。
     - サービス・スーパークラスを作成している場合は、データベース内のすべてのサービス・スーパークラスの間で固有の名前を使用する必要があります。

       サービス・スーパークラスが作成されると、それに関連付けられるデフォルトのサービス・サブクラスが自動的に作成されます。他のサービス・サブクラスは、親サービス・スーパークラスが作成されてからでなければ作成できません。
     - サービス・サブクラスを作成している場合は、サービス・スーパークラス内のすべてのサービス・サブクラスの間で固有の名前を使用する必要があります。サービス・サブクラスをサービス・スーパークラスと同じ名前にすることはできません。
   - サービス・サブクラスを作成している場合は、親サービス・スーパークラスの名前を指定。特定のサービス・スーパークラスの下に作成されたサービス・サブクラスは、別のサービス・スーパークラスには関連付けることができません。

- オプション: サービス・クラスのエージェント優先順位を指定。エージェント優先順位が DEFAULT に設定されている場合、サービス・クラス内のエージェントには、オペレーティング・システムがすべての DB2 スレッドに割り当てるのと同じ優先順位が割り当てられます。AGENT PRIORITY パラメーターに DEFAULT 以外の値を設定した場合は、デフォルトの優先順位に、次のアクティビティー開始時点の設定値を加えた値に等しい優先順位が、エージェント・スレッドの優先順位として設定されます。例えば、デフォルトの優先順位が 20 でエージェント優先順位を -10 に設定した場合、結果的にエージェントの優先順位は 20 + (-10) = 10 に設定されます。

  エージェント優先順位 DEFAULT は、数値にすると -32768 です。

  Linux® および UNIX® では、有効な値は -32768、-20 から 20 です (負の値は、相対的により高い優先順位を示します)。Windows ベースのプラットフォームでは、有効な値は -32768、-6 から 6 です (負の値は、相対的により低い優先順位を示します)。
- DB2 サービス・クラスを AIX サービス・クラスに関連付ける場合は、アウトバウンド相関関係子ストリングを指定。NULL 値は、外部 WLM サービス・クラスの関連付けがないことを示します。

  アウトバウンド相関関係子が設定されると、次のアクティビティーが開始されるときは、DB2 サービス・クラス内のすべてのスレッドがアウトバウンド相関関係子を使用してオペレーティング・システムのワークロード・マネージャーに関連付けられます。

  サービス・サブクラスでアウトバウンド相関関係子が NONE に設定されていて、関連するサービス・スーパークラスにアウトバウンド相関関係子が指定されている場合、サービス・サブクラスは親サービス・スーパークラスで指定されているアウトバウンド相関関係子を継承します。
- プリフェッチ優先順位を指定します。サービス・クラス内のエージェントがプリフェッチ要求をサブミットできる優先順位を指定できます。指定された値に応じて、プリフェッチ要求は優先順位がHIGH、MEDIUM、LOWのプリフェッチ・キューにルーティングされます。デフォルトのプリフェッチ優先順位は MEDIUM です。
- 収集するアクティビティー・データを指定。アクティビティー・データの収集が使用可能になっている場合、アクティビティーに関する情報は、アクティビティーの終了時にコーディネーター・パーティションから該当するイベント・モニターに送信されます。望むなら、アクティビティーが実行されたすべてのデータベース・パーティションから、実行されたステートメント、そのコンパイル環境、および適用可能なすべての入力データ値に関する情報などのデータをイベント・モニターに書き出すことができます。データ・アクティビティーをまったく収集しないように指定することも可能です。デフォルトでは、アクティビティー・データは収集されません。
- 収集された集約アクティビティー情報を指定。サービス・クラスに使用される集約アクティビティー情報は、サービス・クラスの変更操作がコミットされた後にのみ変更されます。
- COLLECT AGGREGATE ACTIVITY DATA を使用した集約アクティビティー・データ収集、または COLLECT AGGREGATE REQUEST DATA を使用

した集約要求データ収集を使用可能にしているサービス・サブクラスが使用するヒストグラム・テンプレートを変更するかどうか。サービス・サブクラスが使用するヒストグラム・テンプレートを更新すると、サービス・クラスまたはワーク・アクションが参照するヒストグラム・テンプレートを表示する SYSCAT.HISTOGRAMTEMPLATEUSE ビューの対応する行が更新されます。ヒストグラムおよびヒストグラム・テンプレートについて詳しくは、140 ページの『ワークロード管理のヒストグラム』を参照してください。

- サービス・クラスを使用可能にするか使用不可にするかを指定。
  - サービス・クラスが使用可能として作成された場合 (デフォルト) は、そのサービス・クラスに接続とアクティビティーをマップすることができます。サービス・クラスが使用不可として作成された場合は、そのサービス・クラスへの新規接続およびアクティビティーのマップが拒否されます。
  - サービス・スーパークラスを使用不可として作成すると、そのサービス・スーパークラスに関連付けられているすべてのサービス・サブクラスは、SYSCAT.SERVICECLASSES ビューを照会したときに使用可能として表示されていたとしても、使用不可の動作をします。

2. 変更をコミットします。 変更をコミットすると、サービス・クラスが SYSCAT.SERVICECLASSES ビューに追加されます。

サービス・クラスの変更

サービス・クラスの定義を変更する場合は、ALTER SERVICE CLASS ステートメントを使用します。

サービス・クラスを変更するためには、DBADM、SYSADM、または SYSCNTRL 権限が必要です。

前提条件について詳しくは、313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』を参照してください。

既にリソースを獲得済みで、実行中のアクティビティーは、ALTER ステートメントの影響を受けません。これらのアクティビティーは、獲得したリソースを保持し、完了まで実行されます。ただし、ALTER SERVICE CLASS 操作の途中でリモート・データベース・パーティションにサブエージェント要求が送信されると、コーディネーター・エージェントとサブエージェントから見えるサービス・クラス定義に違いが発生する可能性があります。次の例について考えてみます。この例では、サービス・クラスのプリフェッチ優先順位が初めは MEDIUM に設定されています。

表 8. コーディネーター・エージェントとサブエージェントの間で生じる、変更されたサービス・クラスの見え方の違い

イベントの順序	接続 1	接続 2
1	コーディネーター・エージェントがリモート・パーティションに DSS 要求を送信する (サービス・クラスのプリフェッチ優先順位は事前に MEDIUM に設定されていた)	

表 8. コーディネーター・エージェントとサブエージェントの間で生じる、変更されたサービス・クラスの見え方の違い (続き)

イベントの順序	接続 1	接続 2
2		ALTER SERVICE CLASS が発行され、プリフェッチ優先順位が HIGH に設定される
3		COMMIT が発行される (変更されたサービス・クラスのプロパティーはカタログ・パーティションでコミットされ、すべてのデータベース・パーティションでメモリーにロードされる)
4	リモート・サブエージェントが DSS 要求を受け取る。この時点になって、サブエージェントはサービス・クラス定義の新しいプリフェッチ優先順位 HIGH を確認する	

上の表で説明した状態は一時的な状態であり、この状態の影響を受けるのは ALTER SERVICE CLASS 操作中にサブエージェント要求を発行する接続だけです。新しい接続はすべて、プリフェッチ優先順位が HIGH になった更新後のサービス・クラス定義を認識します。

サービス・クラスを変更するには、次のようにします。

1. ALTER SERVICE CLASS ステートメントに、以下に挙げるサービス・クラスのプロパティーを 1 つ以上指定します。
   - サービス・クラスを使用可能にするか使用不可にするかを指定。使用可能のサービス・クラスを使用不可に変更した場合、既存の接続やアクティビティーはサービス・クラスに残り、使用不可になる前に割り振られたリソースを完了まで使用します。サービス・クラスに送られる作業がシステムを圧迫している場合や、サービス・クラスに送られるすべての作業を拒否したい場合には、サービス・クラスを使用不可にできます。

     サービス・スーパークラスが使用不可に設定されると、以下のことが起こります。

     a. サービス・スーパークラスが使用不可になります。

     b. そのスーパークラスのサービス・サブクラスが使用不可になります。

     サービス・サブクラスは、親サービス・スーパークラスが使用不可になっている間だけ使用不可になります。サービス・スーパークラスが使用可能になると、サービス・サブクラスは、カタログ表で定義された以前の状態に戻ります。

     サービス・サブクラスが使用不可になっても、親サービス・スーパークラスは影響を受けません。また、同じサービス・スーパークラスに関連付けられている他のサービス・サブクラスも影響を受けません。

デフォルトのサービス・サブクラスは、明示的に使用不可にすることはできません。デフォルトのサービス・サブクラスで新しい要求が実行されないようにするためには、関連付けられているサービス・スーパークラスを使用不可にする必要があります。

- サービス・クラスのエージェント優先順位を指定。エージェント優先順位が DEFAULT に設定されている場合、サービス・クラス内のエージェントには、オペレーティング・システムがすべての DB2 スレッドに割り当てるのと同じ優先順位が割り当てられます。AGENT PRIORITY パラメーターに DEFAULT 以外の値を設定した場合は、デフォルトの優先順位に、次のアクティビティー開始時点の設定値を加えた値に等しい優先順位が、エージェント・スレッドの優先順位として設定されます。例えば、デフォルトの優先順位が 20 でエージェント優先順位を -10 に設定した場合、結果的にエージェントの優先順位は 20 + (-10) = 10 に設定されます。

  エージェント優先順位 DEFAULT は、数値にすると -32768 です。

  Linux および UNIX では、有効な値は -32768、-20 から 20 です (負の値は、相対的により高い優先順位を示します)。Windows ベースのプラットフォームでは、有効な値は -32768、-6 から 6 です (負の値は、相対的により低い優先順位を示します)。
- プリフェッチ優先順位を指定。サービス・クラス内のエージェントがプリフェッチ要求をサブミットできる優先順位を指定できます。指定された値に応じて、プリフェッチ要求は優先順位がHIGH、MEDIUM、LOWのプリフェッチ・キューにルーティングされます。デフォルトのプリフェッチ優先順位は MEDIUM です。プリフェッチ要求がサブミットされた後にプリフェッチ優先順位が変更された場合、その要求の優先順位は変更されません。
- DB2 サービス・クラスを AIX サービス・クラスに関連付ける場合は、アウトバウンド相関関係子ストリングを指定。NULL 値は、外部 WLM サービス・クラスの関連付けがないことを示します。

  アウトバウンド相関関係子が NULL 以外の値から NULL 値に変更されると、次のアクティビティーが開始されるときに、DB2 サービス・クラス内のすべてのスレッドでオペレーティング・システムのワークロード・マネージャーとの関連付けが解除されます。

  サービス・サブクラスでアウトバウンド相関関係子が NONE に設定されていて、関連するサービス・スーパークラスにアウトバウンド相関関係子が指定されている場合、サービス・サブクラスは親サービス・スーパークラスで指定されているアウトバウンド相関関係子を継承します。

  サービス・スーパークラスでアウトバウンド相関関係子を使用する場合は、サービス・スーパークラスのエージェント優先順位をデフォルトに設定する必要があります。

  サービス・サブクラスで (サービス・サブクラス定義の一部として明示的に、またはサービス・スーパークラスから継承することによって暗黙的に) アウトバウンド相関関係子を使用する場合は、サービス・サブクラスのエージェント優先順位をデフォルトに設定する必要があります。

- 収集するアクティビティー・データを指定。アクティビティー・データの収集が使用可能になっている場合、アクティビティーに関する情報は、アクティビティーの終了時にコーディネーター・パーティションから該当するイベント・モニターに送信されます。望むなら、アクティビティーが実行されたすべてのデータベース・パーティションから、実行されたステートメント、そのコンパイル環境、および適用可能なすべての入力データ値に関する情報などのデータをイベント・モニターに書き出すことができます。データ・アクティビティーをまったく収集しないように指定することも可能です。デフォルトでは、アクティビティー・データは収集されません。
- 収集された集約アクティビティー情報を指定。サービス・クラスに使用される集約アクティビティー情報は、サービス・クラスの変更操作がコミットされた後にのみ変更されます。
- COLLECT AGGREGATE ACTIVITY DATA を使用した集約アクティビティー・データ収集、または COLLECT AGGREGATE REQUEST DATA を使用した集約要求データ収集を使用可能にしているサービス・サブクラスが使用するヒストグラム・テンプレートを変更するかどうか。サービス・サブクラスが使用するヒストグラム・テンプレートを更新すると、サービス・クラスまたはワーク・アクションが参照するヒストグラム・テンプレートを表示する SYSCAT.HISTOGRAMTEMPLATEUSE ビューの対応する行が更新されます。ヒストグラムおよびヒストグラム・テンプレートについて詳しくは、 140 ページの『ワークロード管理のヒストグラム』を参照してください。

2. 変更をコミットします。 変更をコミットすると、サービス・クラスが SYSCAT.SERVICECLASSES ビューで更新されます。

サービス・クラスのドロップ

DDL ステートメント DROP SERVICE CLASS を使用してサービス・クラスをドロップします。

サービス・クラスをドロップするためには、DBADM または SYSADM 権限が必要です。

前提条件について詳しくは、 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』を参照してください。

サービス・クラスのドロップには、以下の制約事項があります。

- デフォルトのサービス・スーパークラス (SYSDEFAULTUSERCLASS、SYSDEFAULTMAINTENANCECLASS、および SYSDEFAULTSYSTEMCLASS) およびそれに関連付けられているサービス・サブクラスはドロップできません。デフォルトのサービス・スーパークラスや関連するサービス・サブクラスをドロップする唯一の方法は、データベースをドロップすることです。
- サービス・クラスが以下の条件のいずれかに該当する場合、そのサービス・クラスはドロップできません。
  - 使用可能になっている
  - 何らかのワークロード、ワーク・アクション、またはしきい値によって参照されている

– 現時点で何らかの接続またはアクティビティーがそのサービス・クラスにマップされている

サービス・クラスをドロップするには、次のようにします。

1. ALTER SERVICE CLASS ステートメントを使用して、サービス・クラスを使用不可にします。 サービス・スーパークラスをドロップしている場合は、このアクションによって、そのサービス・スーパークラスに関連付けられているすべてのサービス・サブクラスが使用不可になります。サービス・クラスを使用不可にすると、そのサービス・クラスにはそれ以上アクティビティーを関連付けることができなくなります。サービス・クラスを使用不可にした後、COMMIT ステートメントを発行します。

   サービス・クラスで既に実行中のアクティビティーは、継続して実行されます。現在サービス・クラスにマップされているエージェントは、WLM_GET_SERVICE_CLASS_AGENTS 表関数を使用してリストすることができます。これらのアクティビティーを完了させることを希望しない場合は、表関数によって戻されたアプリケーション ID と FORCE APPLICATION コマンドを使用して、アプリケーションをデータベースから強制終了させることができます。
2. DROP WORKLOAD ステートメントを使用して、サービス・クラスに関連付けられているワークロードをすべてドロップします。各ワークロードをドロップした後、COMMIT ステートメントを発行します。
3. ドロップするサービス・クラスに関連付けられている適用可能なワーク・アクションをすべてドロップします。
   - サービス・スーパークラスをドロップしている場合で、そのサービス・スーパークラスにワーク・アクション・セットが関連付けられている場合は、DROP WORK ACTION SET ステートメントを使用してワーク・アクション・セットをドロップしてください。ワーク・アクション・セットをドロップした後、COMMIT ステートメントを発行します。
   - サービス・サブクラスをドロップしている場合で、そのサービス・サブクラスにワーク・アクションがマップされている場合は、ALTER WORK ACTION SET ステートメントの DROP WORK ACTION キーワードを使用してワーク・アクションをドロップしてください。あるいは別の方法として、DROP WORK ACTION SET ステートメントを使用し、サービス・サブクラスにマップしているワーク・アクションが含まれるワーク・アクション・セットをドロップすることもできます。各ワーク・アクションをドロップした後、またはワーク・アクション・セットをドロップした後、COMMIT ステートメントを発行します。
4. DROP THRESHOLD ステートメントを使用して、ドロップするサービス・クラスに関連付けられているしきい値をすべてドロップします。各しきい値をドロップした後、COMMIT ステートメントを発行します。
5. ドロップするオブジェクトに応じて、以下を行います。
   - サービス・サブクラスをドロップする場合は、DROP SERVICE CLASS ステートメントを使用してサービス・サブクラスをドロップします。
   - サービス・スーパークラスをドロップする場合は、DROP SERVICE CLASS ステートメントを使用して、そのサービス・スーパークラスに関連付けられているすべてのサービス・サブクラスをドロップし、各サービス・サブクラスが

ドロップされた後で COMMIT ステートメントを発行します。次いで、DROP SERVICE CLASS ステートメントを発行してサービス・スーパークラスをドロップします。

注: サービス・スーパークラスのデフォルトのサービス・サブクラスは、手動ではドロップできません。サービス・スーパークラスのデフォルトのサービス・サブクラスは、サービス・スーパークラスをドロップするときにドロップされます。

6. 変更をコミットします。 変更をコミットすると、サービス・クラスが SYSCAT.SERVICECLASSES ビューから除去されます。

第 3 章 ワークロード

ワークロードとは、ユーザーが定義するエンティティーであり、データベース接続属性に従って、サブミットされたデータベース作業をそのソースに基づいて識別し、後で管理できるようにします。ワークロードを使用して、サービス・スーパークラスまたはサービス・スーパークラスのサービス・サブクラスに作業を割り当てることができます。

ワークロード・オブジェクトは以下の項目で構成されます。

- データベース内で固有のワークロード名。
- ワークロードの固有の整数 ID。データ・サーバーによって内部で生成・使用されます。
- データベース接続またはアプリケーションとワークロードを関連付けるために満たす必要があるデータベース接続属性。
- 以下の項目を含むワークロード属性。
  - ワークロードの割り当て先となる DB2 サービス・クラスの名前。
    サービス・クラス名を指定しない場合、ワークロードはデフォルトのユーザー・サービス・クラス SYSDEFAULTUSERCLASS にマップされます。
  - ワークロードのオカレンスがデータベースへのアクセスを許可されるかどうかを示す値。
    デフォルトでは、ワークロード・オカレンスはデータベースにアクセスできます。詳しくは、57 ページの『ワークロード・オカレンスにデータベースへのアクセスを許可する』および 58 ページの『ワークロード・オカレンスにデータベースへのアクセスを許可しない』を参照してください。
  - ワークロードが使用不可かどうかを示す値。
    デフォルトは「使用可能」です。詳しくは、59 ページの『ワークロードを使用可能にする』および 59 ページの『ワークロードを使用不可にする』を参照してください。
- データ・サーバー上の他のワークロードと比較した場合の、ワークロードの評価順序または位置。
  詳しくは、50 ページの『ワークロードの割り当て』を参照してください。

ワークロードは CREATE WORKLOAD ステートメントを使用して作成できます。
ワークロードは ALTER WORKLOAD ステートメントを使用して変更できます。
ワークロードは DROP WORKLOAD ステートメントを使用してドロップできます。

ワークロードは、SYSCAT.WORKLOADS ビューを照会することによって表示できます。
各ワークロードに対して指定した接続属性は、SYSCAT.WORKLOADCONNATTR ビューを照会することによって表示できます。
SYSCAT.WORKLOADAUTH ビューを照会すると、ワークロードを使用する権限がある人物を表示できます。

© Copyright IBM Corp. 2007, 2008

サポートされるデータベース接続属性は次のとおりです。ワークロードにおいて少なくとも 1 つのデータベース接続属性を指定する必要があります。各接続属性は 1 つ以上の値を持つことができます。ワークロード用の特定の接続属性の値が指定されていない場合、データ・サーバーはワークロードの評価時にその属性を調べません。

表 9. ワークロード定義の接続属性

接続属性	説明
アプリケーション名	クライアントで実行中のアプリケーションの名前で、データ・サーバーに認識されます。アプリケーション名は、システム・モニター出力の「アプリケーション名」フィールドに表示される値と等しくなります。詳しくは、**appl_name** モニター・エレメントを参照してください。
システム許可 ID	SYSTEM_USER 特殊レジスターに設定されている、データベースに接続したユーザーの許可 ID。異なる許可 ID を持つユーザーとして接続することにより、SYSTEM_USER の値を変更できます。
セッション許可 ID	SESSION_USER 特殊レジスターに設定されている、アプリケーションの現行セッションで使用される許可 ID。 SET SESSION AUTHORIZATION ステートメントを使用することにより、SESSION_USER の値を変更できます。
セッション許可 ID のグループ	現行セッション・ユーザーが属するグループ。
セッション許可 ID のロール	現行セッション・ユーザーに付与されるロール。詳しくは、以下を参照してください。 • ロール • GRANT ROLE ステートメント • REVOKE ROLE ステートメント
クライアント・ユーザー ID	CURRENT CLIENT_USERID (または CLIENT USERID) 特殊レジスターで設定されている、クライアント情報からのクライアント・ユーザー ID。 sqleseti (クライアント情報の設定) API を使用することによって、クライアント・ユーザー ID の値を変更できます。

表 9. ワークロード定義の接続属性 (続き)

接続属性	説明
クライアント・アプリケーション名	CURRENT CLIENT_APPLNAME (または CLIENT APPLNAME) 特殊レジスターで設定されている、クライアント情報からのアプリケーション名。クライアント・アプリケーション名は、システム・モニター出力の「TP モニター・クライアント・アプリケーション名」フィールドに表示される値と等しくなります。 sqleseti API を使用することによって、クライアント・アプリケーション名の値を変更できます。
クライアント・ワークステーション名	CURRENT CLIENT_WRKSTNNAME (または CLIENT WRKSTNNAME) 特殊レジスターで設定されている、クライアント情報からのワークステーション名。 sqleseti API を使用することによって、クライアント・ワークステーション名の値を変更できます。
クライアント・アカウント情報	CURRENT CLIENT_ACCTNG (または CLIENT ACCTNG) 特殊レジスターで設定されている、クライアント情報からの会計情報ストリング。 sqleseti API を使用することによって、クライアント会計情報ストリングの値を変更できます。

3 層から成るクライアント/サーバー環境では、データベース接続は、アプリケーションサーバーから確立されます。アプリケーション・サーバーは sqleseti API を使用して、クライアント情報を DB2 データ・サーバーに渡すことができます。そうでない場合は、アプリケーション・サーバーに関する情報のみが渡され、その情報は、このアプリケーション・サーバーを介して送付されるすべてのクライアント要求で同じになると思われます。クライアントのユーザー ID、クライアント・アプリケーションの名前、クライアント・ワークステーションの名前、およびクライアント会計情報ストリングなどのクライアント情報属性をワークロード定義に指定することで、さまざまなクライアントで実行するユーザーをさまざまなワークロード (および、さまざまなサービス・クラス) に割り当てることができます。

最初の作業単位の開始時に、データベース接続の接続属性と、ワークロード定義で指定されている接続属性とが一致するなら、そのデータベース接続がワークロードに割り当てられます。接続属性が変更された場合には、次の作業単位の開始時にデータベース接続は異なるワークロードに再割り当てされます。詳しくは、50 ページの『ワークロードの割り当て』を参照してください。

SYSDEFAULTUSERWORKLOAD はデフォルトのワークロードです。定義したどのワークロードにも接続属性がマップされない場合、データベース接続は SYSDEFAULTUSERWORKLOAD に割り当てられ、デフォルトでは SYSDEFAULTUSERCLASS サービス・クラスで作業が実行されます。この状態は、新規データベースまたは新規にマイグレーションされたデータベースで発生します。これは、デフォルトのワークロード以外のワークロードが存在しないためです。

環境の使用特性を分析する際に、CREATE WORKLOAD ステートメントを使用して独自のワークロードを作成し、それらを特定のサービス・クラスにマップすることができます。ワークロードを作成する時には、ワークロードの割り当て中に接続属性を評価するために使用する値、およびワークロードを評価する順番 (他のワークロードに対する相対的な順番) の両方を定義します。複数のワークロードが接続属性と一致する場合があるため、評価順序が変更可能であることにより、一致するどのワークロードを選択するかを決めることが可能になります。セッション・ユーザーがワークロードに対する USAGE 特権があるかどうかによっても、一致するどのワークロードを選択するかが決まります。詳しくは、50 ページの『ワークロードの割り当て』を参照してください。

ワークロード・オカレンスは、ワークロード定義と一致する属性を持つデータベース接続です。ワークロードの割り当てが完了すると、ワークロードのオカレンスはデータ・サーバーで開始され、ワークロード定義で指定されたサービス・クラスで実行されます。このワークロード・オカレンスは、接続が終了するか、あるいは接続属性が変更されるまで続きます。後者の場合、次の作業単位の開始時にワークロードの再評価が行われます。ワークロードの再評価および再割り当ては作業単位の間で行われます。このため、ワークロード・オカレンスは、データ・サーバーで定義済みのワークロードと関連したデータベース接続内の 1 つ以上の作業単位で構成されます。複数のワークロード・オカレンスを、各ワークロードごとにデータ・サーバーで並行して実行できます。

以下の図は、複数の要求を A、B、C、D の順で各ワークロードに照らして評価し、次いで特定のワークロードに割り当てて、当てはまるサービス・クラスで実行する様子を示します。既存のワークロードと一致しない要求は、SYSDEFAULTUSERWORKLOAD ワークロードとマッチングされ、デフォルトでは SYSDEFAULTUSERCLASS サービス・スーパークラスで実行されます。デフォルトの保守クラスおよびデフォルトのシステム・クラスで実行するアクティビティーのタイプについては、27 ページの『デフォルトのサービス・スーパークラスおよびサブクラス』を参照してください。

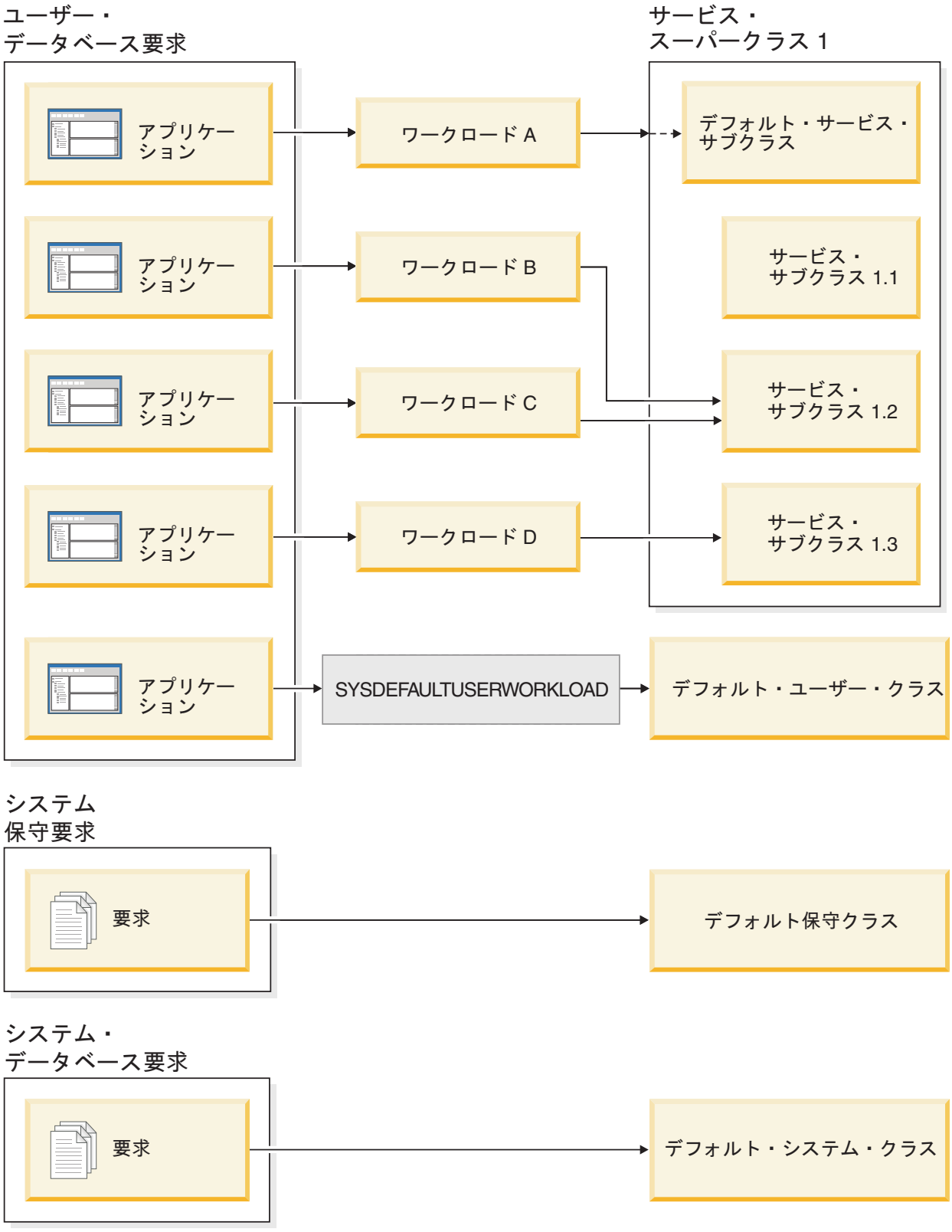


図 13. サービス・クラスとワークロード

ワークロードの割り当て

データベース接続が確立された後の最初の作業単位の初めに、使用可能になっている各ワークロードの接続属性を評価することによって、データ・サーバーは接続をワークロードに割り当てます。

ワークロードが評価される順序は、SYSCAT.WORKLOADS 表にある各ワークロードの EVALUATIONORDER 列値によって決定されます。一致する接続属性を持つワークロードが見つかると、データ・サーバーは、現行のセッション・ユーザーにそのワークロードに対する USAGE 特権があるかどうかを確認します。一致するワークロードに対する USAGE 特権をユーザーが持っている場合、ワークロードの割り当ては完了し、接続はそのワークロードに割り当てられます。一致するワークロードに対する USAGE 特権をユーザーが持っていない場合、データ・サーバーは、セッション・ユーザーが USAGE 特権を持つ、一致するワークロードが見つかるまで、ワークロードの評価を続行します。一致するワークロードが見つからない場合、データ・サーバーは SYSDEFAULTUSERWORKLOAD ワークロードを使用しようとします。現行セッション・ユーザーにそのワークロードに対する USAGE 特権がない場合、SQL4707N が戻され、作業単位は拒否されます。それ以外の場合、接続は SYSDEFAULTUSERWORKLOAD ワークロードに割り当てられます。

CREATE WORKLOAD または ALTER WORKLOAD ステートメントの POSITION キーワードを使用して、評価順序を次のように設定することができます。

- 評価順序内でのワークロードの絶対位置を指定します。次に例を示します。

  ```
  CREATE WORKLOAD...POSITION AT 2
  ```

  POSITION AT 2 は、ワークロードが評価順序の 2 番目に置かれることを意味します。一致するワークロードのうち、評価順序の位置がより高いものが最初に評価されます。つまり、位置 2 と位置 3 にあるワークロードが一致する場合、位置 2 にあるワークロードが位置 3 にあるワークロードの前に評価されます。

  CREATE WORKLOAD または ALTER WORKLOAD ステートメントで指定する位置が既存のワークロードの合計数より大きい場合、ワークロードは、評価順序の最後から 2 番目 (SYSDEFAULTUSERWORKLOAD ワークロードの前) に置かれます。これは、CREATE WORKLOAD または ALTER WORKLOAD ステートメントで POSITION LAST を指定した場合と効果は同じです。
- POSITION BEFORE *workload-name* または POSITION AFTER *workload-name* キーワードを使用します。*workload-name* は既存のワークロードです。このキーワードは、評価順序における新規または変更されたワークロードの位置を、別のワークロードから見た相対的な位置で指定します。次に例を示します。

  ```
  ALTER WORKLOAD...POSITION BEFORE workload2
  ```

POSITION キーワードを指定しない場合は、デフォルトで、新規ワークロードは、評価順序内の他の定義済みのワークロードの後、および常に最後であると見なされる SYSDEFAULTUSERWORKLOAD ワークロードの前に置かれます。

以下のイベントのいずれかが発生したことをデータ・サーバーが検出すると、新しい作業単位の初めにワークロード割り当てが再評価されます。

- 関連した接続属性が変更されている。ワークロード定義で指定可能な接続属性のリストについて詳しくは、45 ページの『第 3 章 ワークロード』の表を参照して

ください。ワークロードの再評価は、現行セッション許可 ID が変更されたときにも行われます。これは、トラステッド・コンテキストが原因でデータベース接続が切り替えられるためです。詳しくは、トラステッド・コンテキストおよびトラステッド接続 (Trusted contexts and trusted connections) を参照してください。

- ワークロードを作成または変更した。
- ユーザー、グループ、またはロールにワークロードに対する USAGE 特権を付与したか、あるいはワークロードに対する USAGE 特権をユーザー、グループ、またはロールから取り消した。

アクティビティーがまだアクティブである間は、別のワークロードに接続を再割り当てできません。アクティビティーは、ロード操作、ストアード・プロシージャーまたは表関数、あるいは WITH HOLD カーソルなどの、複数の UOW 間でリソースを維持する操作です。現在のワークロード・オカレンスは、すべてのアクティビティーが完了するまで実行されます。ワークロードの再割り当ては、次の作業単位の開始時に実行されます。

以下のいずれかの場合、ワークロードの割り当てまたは再割り当てが試行されると、SQL4707N エラーが生じます。

- データ・サーバーが、データベースへのアクセスが禁止されているワークロードに対して接続を割り当てようとした場合。詳しくは、58 ページの『ワークロード・オカレンスにデータベースへのアクセスを許可しない』を参照してください。
- データ・サーバーが SYSDEFAULTUSERWORKLOAD ワークロードに接続を割り当てようとするものの、現行セッション・ユーザーがこのワークロードに対する USAGE 特権を持っていない場合。

DBADM または SYSADM 権限がある場合、データベース接続をデフォルトの管理者ワークロードである SYSDEFAULTADMWORKLOAD ワークロードに割り当てることができます。詳しくは、53 ページの『デフォルト管理ワークロードへの接続の割り当て』を参照してください。

XA トランザクションおよびワークロード再割り当て

XA_END (成功)、XA コミット、および XA ロールバックなどの XA 呼び出しは、作業単位の終わりを示す DB2 COMMIT または ROLLBACK を発行します。ワークロード再評価は作業単位の初めに行うことができるため、これらの XA 呼び出しによってワークロード再評価が開始することがありますが、ワークロード再評価の理由は XA トランザクション自体とは直接関連していません。

デフォルトのワークロード

デフォルトのユーザー・ワークロード SYSDEFAULTUSERWORKLOAD、およびデフォルトの管理ワークロード SYSDEFAULTADMWORKLOAD はデータベースの作成時に作成されます。それらをドロップすることはできません。

バージョン 9.5 以降の DB2 インストール、またはそれへのマイグレーションの後、すべての接続はデフォルトのワークロード SYSDEFAULTUSERWORKLOAD に割り当てられます。このデフォルトのワークロードに属する接続は、デフォルトのユーザー・サービス・スーパークラス SYSDEFAULTUSERCLASS にマップされま

す。接続をデフォルトのワークロードからユーザー定義のワークロードに再マップすることにより、必要に応じて他のユーザー定義のサービス・クラスを使用することができます。さらに、SYSDEFAULTUSERWORKLOAD を変更して、SYSDEFAULTUSERCLASS とは異なるサービス・クラスに接続をマップすることもできます。

SYSDEFAULTUSERWORKLOAD ワークロードは、SYSCAT.WORKLOADS 表を照会することによって表示できます。以下の表では、SYSCAT.WORKLOADS ビューの SYSDEFAULTUSERWORKLOAD ワークロード用の項目を示します。SYSDEFAULTUSERWORKLOAD ワークロードへの接続の割り当て方法については、50 ページの『ワークロードの割り当て』を参照してください。

表 10. SYSCAT.WORKLOADS 内の SYSDEFAULTUSERWORKLOAD 項目

列	値	ALTER WORKLOAD ステートメントを使用して変更可能かどうか (DBADM または SYSADM アクセス権限がある場合)
WORKLOADID	1	不可
WORKLOADNAME	SYSDEFAULTUSERWORKLOAD	不可
EVALUATIONORDER	最後から 2 番目	不可
CREATE_TIME	データベース作成のタイム・スタンプ	不可
ALTER_TIME	最後に行われたワークロード定義の更新のタイム・スタンプ	不可 (ただし、ワークロード定義を更新したときにデータ・サーバーはこの列を変更します)
ENABLED	Y	不可
ALLOWACCESS	Y	可
SERVICECLASSNAME	SYSDEFAULTSUBCLASS	可
PARENTSERVICECLASSNAME	SYSDEFAULTUSERCLASS	可
COLLECTAGGACTDATA	N	不可 (将来の利用のために予約済み)
COLLECTACTDATA	N	可
COLLECTACTPARTITION	C	可
EXTERNALNAME	NULL	不可

詳しくは、SYSCAT.WORKLOADS を参照してください。

SYSDEFAULTADMWORKLOAD ワークロードに割り当てることができるのは、SYSADM または DBADM 権限を持つセッション・ユーザーによってサブミットされる作業単位のみです。このワークロードを使用すると、以下のイベントが発生したときに、SYSADM および DBADM ユーザーはデータベースを照会したり、管理またはモニター・タスクを実行したりできます。

- 管理者が割り当てられているワークロードが、データベースへのアクセスを許可されていない (つまり、ワークロードに対して CREATE WORKLOAD または ALTER WORKLOAD ステートメントの DISALLOW DB ACCESS キーワードが指定されている)。
- しきい値に違反したために管理者がデータベースでの作業を実行できない。

SYSDEFAULTADMWORKLOAD ワークロードは、以下の点で他のワークロードとは異なります。

- ドロップしたり、使用不可にしたりすることはできません。
- DISALLOW DB ACCESS を指定することはできません。
- このワークロードのオカレンスおよびその中のアクティビティーに適用されるしきい値はありません。
- このワークロードは、SYSDEFAULTUSERCLASS サービス・スーパークラスでのみ実行できます。詳しくは、27 ページの『デフォルトのサービス・スーパークラスおよびサブクラス』を参照してください。
- SET WORKLOAD コマンドを使用してのみ、接続をこのワークロードに割り当てることができます。このコマンドは、CLP インターフェースからのみ発行できます。詳しくは、『デフォルト管理ワークロードへの接続の割り当て』を参照してください。

SYSDEFAULTADMWORKLOAD ワークロードは、SYSCAT.WORKLOADS 表を照会することによって表示できます。以下の表では、SYSCAT.WORKLOADS 表の SYSDEFAULTADMWORKLOAD ワークロード用の項目を示します。

表 11. SYSCAT.WORKLOADS 内の SYSDEFAULTADMWORKLOAD 項目

列	値	ALTER WORKLOAD ステートメントを使用して変更可能かどうか (DBADM または SYSADM アクセス権限がある場合)
WORKLOADID	2	不可
WORKLOADNAME	SYSDEFAULTADMWORKLOAD	不可
EVALUATIONORDER	最後	不可
CREATE_TIME	データベース作成のタイム・スタンプ	不可
ALTER_TIME	最後に行われたワークロード定義の更新のタイム・スタンプ	不可 (ただし、ワークロード定義を更新したときにデータ・サーバーはこの列を変更します)
ENABLED	Y	不可
ALLOWACCESS	Y	不可
SERVICECLASSNAME	SYSDEFAULTSUBCLASS	不可
PARENTSERVICECLASSNAME	SYSDEFAULTUSERCLASS	不可
COLLECTAGGACTDATA	N	不可 (将来の利用のために予約済み)
COLLECTACTDATA	N	可
COLLECTACTPARTITION	C	可
EXTERNALNAME	NULL	不可

詳しくは、SYSCAT.WORKLOADS を参照してください。

デフォルト管理ワークロードへの接続の割り当て

SET WORKLOAD コマンドを使用して、デフォルト管理ワークロード SYSDEFAULTADMWORKLOAD に接続を割り当てることができます。

SET WORKLOAD コマンドの使用には特殊権限は必要ありませんが、デフォルト管理ワークロードに接続を割り当てるためには SYSADM または DBADM 権限が必要です。権限がない場合は、ワークロードの割り当て時に SQL0552N が戻されます。

デフォルトの管理ワークロード (SYSDEFAULTADMWORKLOAD) は、DB2 が供給する特殊なワークロード定義で、どの DB2 のしきい値の影響も受けません。このワークロードの目的は、データベース管理者が必要に応じて作業を実施したり、修正アクションを行ったりできるようにすることです。このワークロードはしきい値の影響を受けないため、ワークロード管理制御が制限されており、定期的な日常の作業をサブミットするために使用することはお勧めできません。

デフォルト管理ワークロードに接続を割り当てるには、次のように SET WORKLOAD コマンドを発行します。

```
SET WORKLOAD TO SYSDEFAULTADMWORKLOAD
```

コマンドがいつ有効になるかは、いつコマンドを発行したかによって異なります。

- データベースへ接続する前に SET WORKLOAD TO SYSDEFAULTADMWORKLOAD コマンドを発行した場合は、接続が確立された後、最初の作業単位が開始されるときに、接続が SYSDEFAULTADMWORKLOAD に割り当てられます。
- 作業単位の開始時に SET WORKLOAD TO SYSDEFAULTADMWORKLOAD コマンドを発行した場合は、データベースへの接続が確立された後、sqleseti (クライアント情報設定) 要求以外の最初の要求がサブミットされるときに、接続が SYSDEFAULTADMWORKLOAD に割り当てられます。
- 作業単位の中間で SET WORKLOAD TO SYSDEFAULTADMWORKLOAD コマンドが発行された場合は、接続が確立された後、次の作業単位の開始時に、接続が SYSDEFAULTADMWORKLOAD に割り当てられます。

接続が SYSDEFAULTADMWORKLOAD に関連付けられていると、次のいずれかの状況が発生した場合に、次の作業単位の開始時にワークロードの再割り当てが行われます。

- セッション・ユーザーから SYSADM または DBADM 権限を取り消した。この場合は、SQL0552N が戻されます。
- SET WORKLOAD TO AUTOMATIC コマンドを発行した。このコマンドは、次の作業単位を SYSDEFAULTADMWORKLOAD ワークロードに割り当てず、次の作業単位の開始時には通常のワークロード評価を行うことを示します。詳しくは、50 ページの『ワークロードの割り当て』を参照してください。

ワークロードの作業

ワークロードの作成

CREATE WORKLOAD ステートメントを使用して、カタログにワークロードを追加します。

ワークロードを作成するためには、DBADM または SYSADM 権限が必要です。

前提条件について詳しくは、以下のトピックを参照してください。

- 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』
- 命名規則

ワークロードを作成するには、次のようにします。

1. CREATE WORKLOAD ステートメントを使用して、以下に挙げるワークロードのプロパティーを 1 つ以上指定します。
   - ワークロードの名前。
   - メモリーにキャッシュされる際の、他のワークロードに対するワークロードの相対的な位置。新しいワークロードの位置は、ワークロード割り当て時にワークロードが評価される順序を決定します。デフォルトでは、新しいワークロードは一番後ろに配置されます。これは、そのワークロードが最後 (デフォルトのユーザー・ワークロードが考慮される直前) に評価されることを意味します。詳しくは、 50 ページの『ワークロードの割り当て』を参照してください。
   - 収集するアクティビティー情報のタイプ。デフォルトでは、ワークロードに関連するアクティビティーの情報は、まったくアクティビティー・イベント・モニターに送信されません。
   - 接続属性。一致が生じるためには、ワークロードに指定した接続属性と一致するものを着信接続が提供しなければなりません。詳しくは、 45 ページの『第 3 章 ワークロード』を参照してください。接続属性を指定する際は、値が OR 演算され、属性が AND 演算されることに注意してください。例えば、UserID (bob OR sue OR frank) AND Application (SAS) となります。
   - このワークロードのオカレンスがデータベースへのアクセスを許可されるかどうかを示す値。デフォルトでは、このワークロードのオカレンスはデータベースへのアクセスを許可されます。
   - ワークロードが使用可能か使用不可かを示す値。デフォルトでは、ワークロードは使用可能になっています。
   - このワークロードのオカレンスが実行されるサービス・クラス。デフォルトは SYSDEFAULTUSERCLASS サービス・スーパークラスです。

     ユーザー定義のサービス・スーパークラスを指定して、サービス・スーパークラスの下のユーザー定義のサービス・サブクラスで実行するワークロードをマップしない場合、ワークロード・オカレンスは、サービス・スーパークラスの SYSDEFAULTSUBCLASS サービス・サブクラスで実行します。

     注: SYSDEFAULTUSERCLASS サービス・スーパークラスを含め、サービス・スーパークラスの下に SYSDEFAULTSUBCLASS サービス・サブクラスを指定することはできません。

     ワークロードのオカレンスを SYSDEFAULTSUBCLASS サービス・サブクラスで実行させたくない場合は、このワークロードを通じてユーザー定義のサービス・サブクラスで実行するように、このワークロードをマップすることができます。また、ワーク・アクションを使用して、このワークロードを別のサービス・サブクラスにマップすることもできます (詳しくは 90 ページの『ワーク・アクションとワーク・アクション・セット』を参照)。

2. 変更をコミットします。 変更をコミットすると、ワークロードが SYSCAT.WORKLOADS ビューに追加されます。変更をコミットすると、各アプリケーションの次の作業単位の開始時にワークロードの再評価が行われるようになります。どのワークロードを選択するかによっては、アプリケーションが別のワークロードに再割り当てされることがあります。

ワークロードを作成した後、1 人以上のセッション・ユーザーにそのワークロードに対する USAGE 特権を付与することが必要になる場合もあります。(SYSADM または DBADM 権限を持つセッション・ユーザーには、すべてのワークロードを使用する暗黙特権があります。) 接続の接続属性がワークロードの接続属性と完全に一致する場合でも、セッション・ユーザーがそのワークロードに対する USAGE 特権を持っていない場合は、データ・サーバーは、ワークロードの評価を実行する際に、そのワークロードを考慮しません。詳しくは、 60 ページの『ワークロードに対する USAGE 特権の付与』を参照してください。

ワークロードの変更

ALTER WORKLOAD ステートメントは、カタログ内のワークロードを変更します。

ワークロードを変更するためには、DBADM または SYSADM 権限が必要です。

前提条件について詳しくは、 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』を参照してください。

ワークロードを変更するには、次のようにします。

1. ALTER WORKLOAD ステートメントを使用して、以下に挙げるワークロードのプロパティーを 1 つ以上指定します。
   - 接続属性。ワークロードが SYSDEFAULTUSERWORKLOAD または SYSDEFAULTADMWORKLOAD ワークロードである場合以外は、ワークロード定義に接続属性の追加やドロップができます。一致が生じるためには、ワークロードに指定した接続属性と一致するものを着信接続が提供しなければなりません。詳しくは、 45 ページの『第 3 章 ワークロード』を参照してください。ワークロードの接続属性を確認するには、SYSCAT.WORKLOADCONNATTR ビューを照会します。
   - このワークロードのオカレンスがデータベースへのアクセスを許可されるかどうかを示す値。デフォルトでは、このワークロードのオカレンスはデータベースへのアクセスを許可されます。SYSDEFAULTADMWORKLOAD ワークロードによるデータベースへのアクセスを禁止することはできません。
   - ワークロードが使用可能か使用不可かを示す値。デフォルトでは、ワークロードは使用可能になっています。SYSDEFAULTUSERWORKLOAD ワークロードや SYSDEFAULTADMWORKLOAD ワークロードは使用不可にできません。
   - このワークロードのオカレンスが実行されるサービス・クラス。デフォルトは SYSDEFAULTUSERCLASS サービス・スーパークラスです。ユーザー定義のサービス・スーパークラスを指定する場合は、そのサービス・スーパークラスの下にサービス・サブクラスを指定することができます。SYSDEFAULTUSERCLASS サービス・スーパークラスを含め、サービス・スーパークラスの下に SYSDEFAULTSUBCLASS サブクラスを指定することは

できません。加えて、SYSDEFAULTSYSTEMCLASS または SYSDEFAULTMAINTENANCECLASS サービス・スーパークラスは指定できません。

- ワークロード割り当て時にワークロードが評価される順序を決定する、他のワークロードに対するワークロードの相対的な位置。デフォルトでは、新しいワークロードは一番後ろに配置されます。これは、そのワークロードが最後 (デフォルトのユーザー・ワークロードが考慮される直前) に評価されることを意味します。SYSDEFAULTUSERWORKLOAD または SYSDEFAULTADMWORKLOAD ワークロードの位置は指定できません。詳しくは、 50 ページの『ワークロードの割り当て』を参照してください。
- 収集するアクティビティー情報のタイプ。デフォルトでは、ワークロードに関連するアクティビティーの情報は、まったくアクティビティー・イベント・モニターに送信されません。

2. 変更をコミットします。 変更をコミットすると、ワークロードが SYSCAT.WORKLOADS ビューで更新されます。変更をコミットすると、各アプリケーションの次の作業単位の開始時にワークロードの再評価が行われるようになります。どのワークロードを選択するかによっては、アプリケーションが別のワークロードに再割り当てされることがあります。

ワークロードを変更した後、1 人以上のセッション・ユーザーにそのワークロードに対する USAGE 特権を付与することが必要になる場合もあります。(SYSADM または DBADM 権限を持つセッション・ユーザーには、すべてのワークロードを使用する暗黙特権があります。) 接続の接続属性がワークロードの接続属性と完全に一致する場合でも、セッション・ユーザーがそのワークロードに対する USAGE 特権を持っていない場合は、データ・サーバーは、ワークロードのオカレンスを作成するためにワークロードに接続を関連付けません。詳しくは、 60 ページの『ワークロードに対する USAGE 特権の付与』を参照してください。

ワークロード・オカレンスにデータベースへのアクセスを許可する

データベースへのアクセスを許可していなかったワークロードで、これからオカレンスを実行できるようにする場合は、ワークロードに変更を加えてデータベースへのアクセスが許可されるようにします。デフォルトでは、ワークロードは作成されたときに、データベースへのアクセスを許可されます。

ワークロードを変更してデータベースにアクセスできるようにするためには、DBADM または SYSADM 権限が必要です。

前提条件について詳しくは、 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』を参照してください。

ワークロードにデータベースへのアクセスを許可しないときも、データ・サーバーは引き続き、ワークロード割り当てを実行する際にそのワークロードを審査します。しかし、そのワークロードのオカレンスはすべて拒否されてエラーになります。ワークロードにデータベースへのアクセスを許可するには、次のようにします。

1. ALTER WORKLOAD ステートメントの ALLOW DB ACCESS オプションを使用して、ワークロードにデータベースへのアクセスを許可します。 例えば、WL1 というワークロードにデータベースへのアクセスを許可する場合は、次のステートメントを指定します。

   ```
   ALTER WORKLOAD WL1 ALLOW DB ACCESS
   ```

2. 変更をコミットします。 変更をコミットすると、ワークロードが SYSCAT.WORKLOADS ビューで更新されます。

ワークロードを変更して、ワークロード・オカレンスがデータベースへのアクセスを許可されても、それが有効になるのはデータ・サーバーがそのワークロードの次の作業単位を分析するときです。例えば、DISALLOW DB ACCESS を指定していたワークロード A に変更を加えて ALLOW DB ACCESS を指定すると、ワークロード A の新しいオカレンスは実行を許可されます。変更前は、ワークロード A のすべてのオカレンスが拒否され、エラーになっていました。

ワークロード・オカレンスにデータベースへのアクセスを許可しない

このタスクを使用して、どのワークロードがデータベースにアクセスできるかを制御します。ワークロード・オカレンスが実行できるようになる前に、データ・サーバーは、ワークロードがデータベースへのアクセスを許可されるかどうかを検査します。ワークロード・オカレンスにデータベースへのアクセスを許可しない場合は、ワークロード・オカレンスが拒否されたことを示すエラーが戻されます。

ワークロードによるデータベースへのアクセスを許可しないようにするためには、DBADM または SYSADM 権限が必要です。

前提条件について詳しくは、 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』を参照してください。

ワークロード・オカレンスを許可しないこととワークロードを使用不可にすることとは異なります。ワークロードを使用不可にすると、ワークロード定義はメモリーにキャッシュされないため、ワークロードの割り当ての際に考慮されません。ワークロードにデータベースへのアクセスを許可しないようにするには、次のようにします。

1. 次の例に示すように、ALTER WORKLOAD ステートメントの DISALLOW DB ACCESS オプションを使用します。

   ```
   ALTER WORKLOAD workload-name DISALLOW DB ACCESS ...
   ```

2. 変更をコミットします。 変更をコミットすると、ワークロードが SYSCAT.WORKLOADS ビューで更新されます。

ワークロードに対して行われた、オカレンスにデータベースへのアクセスを許可しないための変更は、既に実行されているワークロード・オカレンスの次の作業単位が開始されるときに有効になります。例えば、 ALLOW DB ACCESS と指定されているワークロード A に対し、DISALLOW DB ACCESS を指定して変更した場合、既に実行されているワークロード A のオカレンスは、次の作業単位が開始されるときに SQL エラーを受け取ります。ワークロード A の新規オカレンスは拒否されます。

ワークロードを使用可能にする

DB2 データ・サーバーは、ワークロードに指定された接続属性を現行セッションの接続属性と突き合わせて検査します。データ・サーバーは、一致するワークロードを探す際、使用不可にされたワークロードを考慮しません。

ワークロードを変更するためには、SYSADM または DBADM 権限が必要です。

前提条件について詳しくは、 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』を参照してください。

ワークロードは、作成時はデフォルトで使用可能になっています。ワークロードを使用不可として作成した場合、ワークロード評価の実行時にデータ・サーバーにそのワークロードを考慮させるには、これを使用可能にする必要があります。

ワークロードを使用可能にするには、次のようにします。

1. 使用可能にするワークロードを識別します。 次の例に示すように、SYSCAT.WORKLOADS ビューを照会することによって、使用不可になっているワークロードのセットを表示できます。

   ```
   SELECT * FROM SYSCAT.WORKLOADS WHERE ENABLED='N'
   ```

2. ALTER WORKLOAD ステートメントを使用して、使用不可になっているワークロードを使用可能にします。

   ```
   ALTER WORKLOAD...ENABLE
   ```

   ALTER WORKLOAD ステートメントが成功すると、ワークロードの定義がデータベース・カタログに書き込まれます。

3. 変更をコミットします。 変更をコミットすると、ワークロードが SYSCAT.WORKLOADS ビューで更新されます。

実際にワークロードが使用可能になるのは次の作業単位が開始されるときです。その時点でワークロードの再評価が行われ、データ・サーバーはワークロードの再評価を実行する際に新しく使用可能になったワークロードを考慮するようになります。

ワークロードを使用不可にする

このタスクを使用して、ワークロードの割り当て中に、特定のワークロードが考慮の対象にならないようにします。DB2 データ・サーバーは、ワークロードに指定された接続属性を現行セッションの接続属性と突き合わせて検査します。ワークロードを使用不可にすると、データ・サーバーは、一致するワークロードを検索する際にそのワークロードを考慮しなくなります。代わりに、データ・サーバーは次の一致するワークロードに作業単位を割り当てます。一致するカスタム定義のワークロードがない場合、その作業単位はデフォルトのワークロードに割り当てられます。

ワークロードを作成したり変更したりするためには、SYSADM または DBADM 権限が必要です。

前提条件について詳しくは、 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』を参照してください。

ワークロードを使用不可にするには、次のようにします。

1. ALTER WORKLOAD ステートメントの DISABLE オプションを使用して、ワークロードを使用不可にします。

   ```
   ALTER WORKLOAD...DISABLE
   ```

2. 変更をコミットします。 変更をコミットすると、ワークロードが SYSCAT.WORKLOADS ビューで更新されます。

実際にワークロードが使用不可になるのは次の作業単位が開始されるときです。その時点でワークロードの再評価が行われ、接続属性が一致し、必要な権限がある、次に使用可能なワークロードに接続が割り当てられます。

ワークロードに対する USAGE 特権の付与

ワークロードを接続に関連付けるには、セッション・ユーザーがそのワークロードに対する USAGE 特権を持っている必要があります。SYSADM および DBADM 権限を持っているユーザーは、暗黙的にすべてのワークロードに対する USAGE 特権を持ちます。

GRANT USAGE ON WORKLOAD ステートメントを使用するためには、SYSADM または DBADM 権限が必要です。

前提条件について詳しくは、 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』を参照してください。

データ・サーバーが着信接続の属性と一致するワークロードを見つけると、データ・サーバーはセッション・ユーザーがそのワークロードに対する USAGE 特権を持っているかどうかを検査します。セッション・ユーザーがそのワークロードに対する USAGE 特権を持っていない場合、データ・サーバーは次の一致するワークロードを探します。(つまり、セッション・ユーザーが USAGE 特権を持っていないワークロードは、あたかも存在していないかのように扱われます。) したがって、ワークロードの USAGE 特権を使用することにより、一致する複数のワークロードの中からどのワークロードにユーザー、グループ、またはロールを割り当てるかを一層制御できるようになります。例えば、同じ接続属性を持つワークロードを複数定義しておいて、それぞれのワークロードに対する USAGE 特権を特定のユーザー、グループ、またはロールだけに付与することができます。詳しくは、 50 ページの『ワークロードの割り当て』を参照してください。

クライアントは、クライアント・ユーザー ID、クライアント・アプリケーション名、クライアント・ワークステーション名、およびクライアント会計情報ストリング (これらはワークロードへの接続の割り当てに使用される接続属性の一部です) を許可なしで設定することができます。それで、ワークロードの USAGE 特権は、どのセッション・ユーザーにワークロードを使用する権限を持たせるかを制御するためにも使用することができます。

USAGE 特権の情報は、SYSCAT.WORKLOADAUTH ビューを照会することによって表示できます。

RESTRICT オプションを使用せずにデータベースを作成する場合は、データベース作成時に、SYSDEFAULTUSERWORKLOAD ワークロードに対する USAGE 特権が PUBLIC に付与されます。RESTRICT オプションを使用してデータベースを作成する場合は、このワークロードに対する USAGE 特権を明示的に非 SYSADM および

非 DBADM ユーザーに付与する必要があります。セッション・ユーザーが SYSDEFAULTUSERWORKLOAD を含むどのワークロードに対しても USAGE 特権を持っていない場合は、データ・サーバーがワークロードにデータベース接続を関連付けようとする際に SQL4707N が戻されます。

ワークロードに対する USAGE 特権を付与するには、次のようにします。

1. GRANT USAGE ON WORKLOAD ステートメントを使用します。 特定のユーザー、グループ、ロール、または PUBLIC に対して USAGE 特権を付与することができます。例えば、ACCOUNTS ワークロードの USAGE 特権を CPA グループに付与するには、次のステートメントを発行します。

   ```
   GRANT USAGE ON WORKLOAD ACCOUNTS TO GROUP CPA
   ```

   SYSDEFAULTADMWORKLOAD ワークロードに対する USAGE 特権は付与できません。SYSDEFAULTADMWORKLOAD ワークロードは、SET WORKLOAD TO SYSDEFAULTADMWORKLOAD コマンドを発行する SYSADM および DBADM ユーザーだけが使用できます。
2. 変更をコミットします。 変更をコミットすると、SYSCAT.WORKLOADAUTH ビューが更新されます。GRANT ステートメントがコミットされるまで、データ・サーバーは、新しく許可されたユーザー、グループ、またはロールに対してワークロード割り当てを実行する際にそのワークロードを考慮できません。

ワークロードに対する USAGE 特権の取り消し

REVOKE USAGE ON WORKLOAD ステートメントを使用して、ワークロードに対する USAGE 特権を取り消すことができます。

REVOKE USAGE ON WORKLOAD ステートメントを使用するためには、SYSADM または DBADM 権限が必要です。

前提条件について詳しくは、 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』を参照してください。

SYSDEFAULTADMWORKLOAD ワークロードに対する USAGE 特権は、明示的に取り消すことはできません。このワークロードは、SET WORKLOAD TO SYSDEFAULTADMWORKLOAD コマンドを発行する SYSADM および DBADM ユーザーだけが使用できます。したがって、REVOKE USAGE ON WORKLOAD ステートメントは、SYSDEFAULTADMWORKLOAD に対しては機能しません。

ワークロードに対する USAGE 特権を取り消すには、次のようにします。

1. REVOKE USAGE ON WORKLOAD ステートメントを使用します。 特定のユーザー、グループ、ロール、または PUBLIC から USAGE 特権を取り消すことができます。例えば、ACCOUNTS ワークロードの USAGE 特権を PUBLIC から取り消すには、次のステートメントを指定します。

   ```
   REVOKE USAGE ON WORKLOAD ACCOUNTS FROM PUBLIC
   ```
2. 変更をコミットします。 変更をコミットすると、SYSCAT.WORKLOADAUTH ビューが更新されます。REVOKE ステートメントがコミットされるまで、データ・サーバーは、ワークロード割り当てを実行する際にこのワークロードを考慮します。

ワークロードのドロップ

ワークロードをドロップすると、ワークロードがデータベース・カタログから除去されます。

ワークロードをドロップするためには、DBADM または SYSADM 権限が必要です。

前提条件について詳しくは、 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』を参照してください。

ワークロードをドロップするには、次のようにします。

1. ALTER WORKLOAD ステートメントを指定して、ワークロードを使用不可にします。 詳しくは、 59 ページの『ワークロードを使用不可にする』を参照してください。ワークロードを使用不可にすると、そのワークロードの新しいオカレンスはデータベースで実行できなくなります。
2. WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数を使用して、このワークロードに実行中のオカレンスがないことを確認します。 詳しくは、WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数を参照してください。

   WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数は、アクティブになっているワークロード・オカレンスに対応するアプリケーション・ハンドルを戻します。FORCE APPLICATION コマンドを使用し、アプリケーション・ハンドルを使用してアプリケーションを終了させることができます。
3. DROP WORKLOAD ステートメントを指定して、ワークロードをドロップします。 例えば、ACCTNG ワークロードをドロップする場合は以下のステートメントを指定します。

   ```
   DROP WORKLOAD ACCTNG
   ```
4. 変更をコミットします。 変更をコミットすると、ワークロードが SYSCAT.WORKLOADS ビューから除去されます。さらに、ワークロードの許可情報が SYSCAT.WORKLOADAUTH ビューから除去されます。

第 4 章 しきい値

しきい値を使用してリソースの誤用を検出すること、またはシステムの過負荷を初期のうちに検出することができます。

しきい値を使用して、ある特定のリソースの使用量に対して明確な限度を設定することができます。しきい値に違反すると、特定のアクションが起動するようにできます。サポートされるアクションは以下のとおりです。

- しきい値に違反する原因となったアクティビティーの処理を停止する (STOP EXECUTION)
- 処理の続行 (CONTINUE)
- しきい値に違反したアクティビティーに関する情報を収集する

しきい値に違反したアクティビティーが停止されているかまたは続行を許可されているかのどちらであっても、アクティビティーに関する詳細情報を収集することができます。しきい値に違反したアクティビティーに関する情報は、アクティビティーの実行完了時に、アクティブな ACTIVITIES イベント・モニターによって収集されます。

しきい値に違反したアクティビティーに関する情報を収集するとしてもしないとしても、しきい値に違反したときは、しきい値違反の記録がアクティブな THRESHOLD VIOLATIONS イベント・モニターに書き込まれます。

しきい値は 2 種類の一般的なカテゴリーに分類されます。

- アクティビティーしきい値。
  アクティビティーしきい値は、個別のアクティビティーに適用されます。例えば、最大アクティビティー合計時間のしきい値は、単一のアクティビティーがデータ・サーバーにとどまることができる合計時間を制限するので、アクティビティーしきい値です。
- 集約しきい値。
  集約しきい値は、複数のアクティビティーのセットから計算された測定に限界を設定します。例えば、サービス・クラス内の並行アクティビティーの最大数は、集約しきい値です。

各しきい値はドメイン に対して適用されます。しきい値のドメインは、しきい値が作用するデータベース・オブジェクトを定義します。しきい値の影響を受ける可能性があるのは、そのドメイン内で起こるアクティビティーのみです。しきい値のドメインは以下のとおりです。

- データベース
- サービス・スーパークラス
- サービス・サブクラス
- ワーク・アクション
- ワークロード

© Copyright IBM Corp. 2007, 2008

これらのしきい値のドメインのそれぞれにおいて、しきい値は単一のワークロード・オカレンス、1 つのデータベース・パーティション、またはデータベースのすべてのパーティションのいずれかに対して強制可能なものとなる場合があります。これは、しきい値の*適用範囲* として知られています。

サービス・クラスの集約しきい値は 2 つの適用範囲、データベースおよびデータベース・パーティションのうちの 1 つを持ちます。データベース・パーティション・レベルで適用される集約しきい値の例は、パーティション上のサービス・スーパークラスに対する同時接続の最大数です。データベース・レベルで (つまり、すべてのデータベース・パーティションにわたって) 適用される集約しきい値の例は、すべてのパーティションにまたがるサービス・クラスの並行アクティビティーの最大数です。

いくつかのしきい値は組み込みキューを持っており、*しきい値境界* とキューイング*境界* という 2 つの境界で定義されます。これらのしきい値はキューイングしきい*値* として知られています。通常、キューイングしきい値のしきい値境界は、一定レベルの並行性を強制します (並行アクティビティーの最大数など)。これを上回る追加の要求についてはキューに入れられます。キューイング境界は、キューの上限を定義します。具体的には、あるアクティビティーがキューイングしきい値のしきい値境界に違反すると、そのしきい値によってトラッキングされる新しい作業要求は、先入れ先出し法で自動的にキューに入れられます。これはキューがキューイング境界によって指定されたサイズに達するまで続きます。キューがいっぱいになって上限に達すると、そのしきい値によってトラッキングされる作業が新しく到着した場合に、しきい値に指定されているアクションがこれに適用されることになります。例えば、STOP EXECUTION のアクションでは新しく到着する作業が拒否されます。

上限を*無制限* と定義することも可能です。この場合、キューのサイズに上限はありません。この状態では、新しく到着する作業はキューに追加されます。上限にハード・リミットを定義し、CONTINUE のアクションをしきい値のアクションとして定義すると、しきい値境界に違反して新しく到着する作業はすべてキューに追加され、しきい値はキューイング境界が無制限であるかのような動作をします。

異なるしきい値は、異なるタイプの作業をトラッキングします。例えば、しきい値は SQL ベースのアクティビティー、ロード・ユーティリティーなどのユーティリティー、接続、ワークロード・オカレンスなどをトラッキングする可能性があります。しきい値の対象となる作業のことを、そのしきい値の*トラッキングされる作業* と言います。例えば、timeron の数に基づいたしきい値は、timeron 値が関連付けられている作業 (この場合は DML ベースのアクティビティー) にしか適用されず、DDL またはユーティリティーなど、他のタイプの作業は含まれません。

しきい値は予測的か反応的かのどちらかになります。

- 予測的しきい値境界は、トラッキングされる作業が実行を開始する前に検査されます。予測的しきい値が違反するかどうか検査するために、データ・サーバーは SQL コンパイラーから使用量の見積もりを入手します。
- 反応的しきい値境界は、トラッキングされる作業の一部が実行されている間に検査されます。制御されるリソースの概算の実行時使用量は、反応的しきい値境界

を評価するのに使用されます。実行時使用量の推定量は連続して入手するのではなく、むしろトラッキングされる作業の存続時間中の、選択された定義済みのチェックポイントで入手します。

しきい値はすべてのステートメントに適用されるわけではありません。例えば、COMMIT、ROLLBACK、SAVEPOINT、および ROLLBACK to SAVEPOINT ステートメントには適用されません。

しきい値は CREATE THRESHOLD ステートメントを使用して作成できます。しきい値は ALTER THRESHOLD ステートメントを使用して変更できます。しきい値は DROP THRESHOLD ステートメントを使用してドロップできます。

しきい値は SYSCAT.THRESHOLDS ビューを照会することによって表示できます。

アクティビティーしきい値と集約しきい値

2 種類のワークロード管理のしきい値がサポートされます。アクティビティーしきい値および集約しきい値です。

アクティビティーしきい値は、個別のアクティビティーに適用されます。個別のアクティビティーのリソースの使用量が、それをトラッキングしているしきい値の上限を超えると、対応するアクションが起動され、そのアクティビティーに 1 回適用されます。適用された後、アクティビティーしきい値はそのアクティビティーに対して非活動化されます。例えば、時間のしきい値を 5 分、そしてこのしきい値に対するアクションを CONTINUE と定義すると仮定します。アクティビティーがこのしきい値に違反する場合、しきい値は 5 分ごとに再適用されるのではなく、1 回だけ適用されます。

集約しきい値は、データベース内の作業の複数のエレメントに対して、集合的な制御を行います。集約しきい値を使用して定義する境界は、合計値として機能し、しきい値によってトラッキングされるあらゆる作業の合計値となります。新しくインスタンス化された作業が原因で上限に違反した場合、対応するアクションが起動します。上限を違反する原因となった作業だけが、トリガー・アクションの影響を受けます。

しきい値のサマリー

このトピックの表は、使用可能なしきい値、およびそれぞれの対応する定義ドメインと適用範囲のクイック・サマリーです。

表 12. 定義ドメインおよび適用範囲ごとのしきい値

	適用範囲: データベース	適用範囲: データベース・パーティション	適用範囲: ワークロード・オカレンス
定義ドメイン: データベース	• 75 ページの『CONCURRENTDBCOORDACTIVITIES しきい値』 • 67 ページの『ESTIMATEDSQLCOST しきい値』 • 68 ページの『SQLROWSRETURNED しきい値』 • 69 ページの『ACTIVITYTOTALTIME しきい値』 • 66 ページの『CONNECTIONIDLETIME しきい値』	• 70 ページの『TOTALDBPARTITIONCONNECTIONS しきい値』 • 67 ページの『SQLTEMPSPACE しきい値』	なし

表 12. 定義ドメインおよび適用範囲ごとのしきい値 (続き)

	適用範囲: データベース	適用範囲: データベース・パーティション	適用範囲: ワークロード・オカレンス
定義ドメイン: ワーク・アクション・セット	• 75 ページの『CONCURRENTDBCOORDACTIVITIES しきい値』 • 67 ページの『ESTIMATEDSQLCOST しきい値』 • 68 ページの『SQLROWSRETURNED しきい値』 • 69 ページの『ACTIVITYTOTALTIME しきい値』	• 67 ページの『SQLTEMPSPACE しきい値』	なし
定義ドメイン: サービス・スーパークラス	• 75 ページの『CONCURRENTDBCOORDACTIVITIES しきい値』 • 67 ページの『ESTIMATEDSQLCOST しきい値』 • 68 ページの『SQLROWSRETURNED しきい値』 • 69 ページの『ACTIVITYTOTALTIME しきい値』 • 『CONNECTIONIDLETIME しきい値』	• 71 ページの『TOTALSCPARTITIONCONNECTIONS しきい値』 • 67 ページの『SQLTEMPSPACE しきい値』	なし
定義ドメイン: サービス・サブクラス	• 75 ページの『CONCURRENTDBCOORDACTIVITIES しきい値』 • 67 ページの『ESTIMATEDSQLCOST しきい値』 • 68 ページの『SQLROWSRETURNED しきい値』 • 69 ページの『ACTIVITYTOTALTIME しきい値』	• 67 ページの『SQLTEMPSPACE しきい値』	なし
定義ドメイン: ワークロード	なし	• 72 ページの『CONCURRENTWORKLOADOCCURRENCES しきい値』	• 73 ページの『CONCURRENTWORKLOADACTIVITIES しきい値』

アクティビティーしきい値

CONNECTIONIDLETIME しきい値

CONNECTIONIDLETIME しきい値は、接続をアイドル (つまり、ユーザー要求を処理していない状態) にしておくことのできる時間の長さの上限を指定します。

タイプ アクティビティー

定義ドメイン
データベースまたはサービス・スーパークラス

適用範囲
データベース

トラッキングされる作業
ユーザー接続

キューイング
不可

単位 分、時間、または日数で表現される時間の長さ

予測的か反応的か
反応的

しきい値で指定された時間よりも長く接続がアイドル状態のままになっており、しきい値アクションが STOP EXECUTION である場合、接続は閉じられます。

このしきい値の細分度は 5 分です。したがって、このしきい値に指定される値はすべて、最も近いゼロ以外の 5 (分) の倍数に丸められます。

ESTIMATEDSQLCOST しきい値

ESTIMATEDSQLCOST しきい値は、DML アクティビティーで許可される見積コストの最大値を指定します。

タイプ アクティビティー

定義ドメイン
データベース、サービス・スーパークラス、サービス・サブクラス、ワーク・アクション

適用範囲
データベース

トラッキングされる作業
このトピックで後述される情報を参照

キューイング
不可

単位 timeron で表現される見積 SQL コスト

予測的か反応的か
予測的

このしきい値では、以下のアクティビティーがトラッキングされます。

- コーディネーター・パーティションで発行される DML アクティビティー。
- ユーザー・アプリケーションから呼び出されるネストされた DML アクティビティー。したがって、DB2 ユーティリティー、SYSPROC ストアード・プロシージャー、内部 SQL などの中から発行される DML など、データ・サーバーによって内部的に発行される DML アクティビティーは、このしきい値の影響を受けません。ただし、アクティビティーのコストが親アクティビティーの見積もりに含まれている場合は例外です。この場合は、アクティビティーは間接的にトラッキングされます。間接的にトラッキングされるアクティビティーの例としては、トリガーが挙げられます。

DML 作業タイプをもつワーク・クラスに分類されるアクティビティーについては、94 ページの『ワーク・クラスの作業タイプおよび SQL ステートメント』を参照してください。

データ・サーバーは、IMPORT、EXPORT、およびその他の各 CLP コマンドをユーザー・ロジックとみなします。 IMPORT、EXPORT、およびその他の各 CLP コマンドの中から呼び出されるアクティビティーは、しきい値の影響を受けます。

SQLTEMPSPACE しきい値

SQLTEMPSPACE しきい値は、あらゆるデータベース・パーティションにおいて DML アクティビティーが消費できる TEMPORARY 表スペースの最大量を指定します。 DML アクティビティーでは、しばしば、ソートや中間結果セットの処理などの操作に TEMPORARY 表スペースが使用されます。

タイプ アクティビティー

定義ドメイン
データベース、サービス・スーパークラス、サービス・サブクラス、ワーク・アクション

適用範囲
データベース・パーティション

トラッキングされる作業
このトピックで後述される情報を参照

キューイング
不可

単位 キロバイト (KB)、メガバイト (MB)、またはギガバイト (GB) で表現される TEMPORARY 表スペースの量

予測的か反応的か
反応的

このしきい値では、以下のアクティビティーがトラッキングされます。

- コーディネーター・パーティションで発行される DML アクティビティー。
- ユーザー・アプリケーションから派生するネストされた DML アクティビティー。したがって、DB2 ロジック (ユーティリティー、SYSPROC プロシージャー、または内部 SQL) によって発行される DML アクティビティーはこのしきい値の影響を受けません。

データ・サーバーは、IMPORT、EXPORT、およびその他の各 CLP コマンドをユーザー・ロジックとみなします。 IMPORT、EXPORT、およびその他の各 CLP コマンドの中から呼び出されるアクティビティーは、しきい値の影響を受けます。

SQLROWSRETURNED しきい値

SQLROWSRETURNED しきい値は、データ・サーバーがクライアントに戻すことができる行の最大数を指定します。

タイプ アクティビティー

定義ドメイン
データベース、サービス・スーパークラス、サービス・サブクラス、ワーク・アクション

適用範囲
データベース

トラッキングされる作業
このトピックで後述される情報を参照

キューイング
不可

単位 行の数

予測的か反応的か
反応的

CALL ステートメントから複数の結果セットが戻される場合、しきい値は、全結果セットから戻される行の総数の集約に対してではなく、結果セットごとに別個に適

用されます。例えば、しきい値を 20 行として定義している場合に、CALL ステートメントからそれぞれ 15 行と 19 行を戻す 2 つの結果セットが戻されたとしても、しきい値はトリガーされません。

このしきい値では、以下のアクティビティーがトラッキングされます。

- コーディネーター・パーティションで発行される DML アクティビティー。
- ユーザー・アプリケーションから呼び出されるネストされた DML アクティビティー。したがって、DB2 ユーティリティー、SYSPROC ストアード・プロシージャー、内部 SQL などの中から発行される DML など、データ・サーバーによって内部的に発行される DML アクティビティーは、このしきい値の影響を受けません。

データ・サーバーは、IMPORT、EXPORT、およびその他の各 CLP コマンドをユーザー・ロジックとみなします。 IMPORT、EXPORT、およびその他の各 CLP コマンドの中から呼び出されるアクティビティーは、しきい値の影響を受けます。

ACTIVITYTOTALTIME しきい値

ACTIVITYTOTALTIME しきい値は、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上限を指定します。

タイプ アクティビティー

定義ドメイン
データベース、サービス・スーパークラス、サービス・サブクラス、ワーク・アクション

適用範囲
データベース

トラッキングされる作業
認識されているコーディネーター・アクティビティーおよびネストされたアクティビティー (21 ページの『アクティビティー』を参照)

キューイング
不可

単位 分、時間、または日数で表現される時間の長さ

予測的か反応的か
反応的

アクティビティーがキューイングしきい値によってキューに入れられる場合、アクティビティーの合計時間には、アクティビティーがキューで実行を待っている時間も含まれます。カーソルが開かれている場合は、そのカーソルに関連付けられているアクティビティーはカーソルが閉じられるまで続きます。

このしきい値の細分度は 5 分です。したがって、このしきい値に指定される値はすべて、最も近いゼロ以外の 5 (分) の倍数に丸められます。

時間のしきい値がストアード・プロシージャーに適用される際は、そのストアード・プロシージャーの内部で行われている作業にもしきい値が適用されます。したがって、ストアード・プロシージャーの時間のしきい値が限度に達したときは、そのストアード・プロシージャーの内部で行われているすべての作業が停止します。

最も深いネスト・レベルのアクティビティー実行に適用される時間しきい値の階層は、ストアード・プロシージャーの呼び出しの階層から導き出すことができます。常に、その階層内で最も制限の高い時間しきい値 (つまり、期限が最も近い時間しきい値) が適用されます。

データ・サーバーは、IMPORT、EXPORT、およびその他の各 CLP コマンドをユーザー・ロジックとみなします。 IMPORT、EXPORT、およびその他の各 CLP コマンドの中から呼び出されるアクティビティーは、しきい値の影響を受けます。

アクティビティーしきい値の有効範囲の解決

アクティビティーしきい値は個々のアクティビティーに適用されるため、1 つのアクティビティーに複数のしきい値が適用される場合は、どのしきい値を実施するかを決定する必要があります。集約しきい値では、同じアクティビティーが複数の集約に同時に寄与することができるため (例えば、並行性しきい値で起こるように)、この問題は起きません。

実行中のアクティビティーに適用するためのアクティビティーしきい値の解決方法は、ローカル側のドメイン内で定義された値は、より広域またはよりグローバルなドメインからのあらゆる値をオーバーライドする、という規則に従います。以下にドメインの階層を、ローカルなものからグローバルなものへ向かう順番で示します。

1. ワークロード
2. サービス・サブクラス
3. サービス・スーパークラス
4. ワーク・アクション
5. データベース

一例として、データベースのドメインで定義されたすべてのデータベース照会に対する 1 時間という最大実行時間は、サービス・スーパークラス LARGE QUERIES に対する 5 時間という最大実行時間にオーバーライドされる可能性があり、それもサービス・サブクラス VERY LARGE QUERIES に対する 10 時間という最大実行時間にオーバーライドされる可能性があります。同様に、データベースのドメインで定義された 1 時間という最大実行時間は、迅速に完了する重要なクエリー向けの異なるサービス・スーパークラス内の 10 分間という値にオーバーライドされる可能性があります。

集約しきい値

TOTALDBPARTITIONCONNECTIONS しきい値

TOTALDBPARTITIONCONNECTIONS しきい値は、データベースのコーディネーター・パーティションでの並行データベース接続の最大数を指定します。つまり、このしきい値は、各データベース・パーティションでデータベースに接続できるクライアントの最大数を制御します。このしきい値は、DBADM 権限を持つユーザーには強制されません。

タイプ 集約

定義ドメイン
データベース

適用範囲
データベース・パーティション

トラッキングされる作業
接続

キューイング
可 (0 で強制)

単位 同時接続の数

予測的か反応的か
予測的

例えば、TOTALDBPARTITIONCONNECTIONS しきい値を 10 に設定していて、データベースに 5 つのパーティションがある場合、各パーティションでは 10 (データベース全体では合計で 50) までクライアントを同時接続させることができます。

TOTALDBPARTITIONCONNECTIONS しきい値で制御されるのはコーディネーター接続だけです。サブエージェントによる接続はしきい値のカウントに含まれません。

このしきい値は、同一インスタンス内で複数のデータベースを使用する場合に役立ちます。データベース・パーティションに TotalDBPartitionConnections しきい値を設定することにより、1 つのデータベースからのクライアント接続がデータベース・パーティションで使用可能な接続をすべて使用してしまうことのないようにすることができます。

max_connections データベース・マネージャー構成パラメーターは、データベース全体で使用する接続の最大数に対応できる大きさに設定してください。データベースに TOTALDBPARTITIONCONNECTIONS しきい値を設定する場合は、**max_connections** をしきい値以上の値に設定する必要があります。同一インスタンス上で複数のデータベースを実行させる場合は、**max_connections** を、全データベースの接続の最大数に対応できる大きさに設定するようにしてください。データ・サーバーでは、同時にアクティブにされるデータベースの数をあらかじめ知ることができないため、この条件は検査されません。

TOTALSCPARTITIONCONNECTIONS しきい値

TOTALSCPARTITIONCONNECTIONS しきい値は、サービス・スーパークラスのコーディネーター・パーティションでの並行データベース接続の最大数を指定します。

タイプ 集約

定義ドメイン
サービス・スーパークラス

適用範囲
データベース・パーティション

トラッキングされる作業
接続

キューイング
可

単位 サービス・クラス内での同時接続の数

予測的か反応的か
予測的

サービス・クラス内で接続数が TOTALSCPARTITIONCONNECTIONS しきい値に達すると、それよりも後にサービス・スーパークラスに加わるコーディネーター接続はキューに入れられます。これはキューが指定されたキュー・サイズに達するまで続きます。デフォルトのキュー・サイズはゼロで、これは接続がキューに入らないことを意味します。接続が TOTALSCPARTITIONCONNECTIONS しきい値のキューに加わった場合、その接続は*過渡* 状態にあると見なされます。

トラッキングされる接続には、新しいクライアント接続と、別のサービス・クラスからそのサービス・クラスに切り替わる既存のクライアント接続の両方が含まれます。接続のサービス・クラスの切り替えは、別のサービス・クラスにマップされている別のワークロード定義に関連付けることによって行われます。ワークロードの再評価はトランザクション境界でのみ行われるため、接続のサービス・クラスの切り替えはトランザクション境界でのみ行うことができます。ただし、WITH HOLD カーソルに関連付けられているリソースはトランザクション境界を越えて維持されるため、開かれている WITH HOLD カーソルがある接続では、サービス・スーパークラスの切り替えができません。接続コンセントレーターがオンになっている場合、切り替えられたアプリケーションはすべてサービス・クラスを離れ、チケットを戻します。このチケットは TOTALSCPARTITIONCONNECTIONS しきい値によって保持されます。後続のステートメントでそのアプリケーションへの切り替えがあるときは、アプリケーションは再度サービス・クラスに加わり、そこでしきい値を渡す必要があります。

キュー・サイズがしきい値に達すると、しきい値アクションがトリガーされます。TOTALSCPARTITIONCONNECTIONS しきい値で制御されるのはコーディネーター接続だけです。サブエージェントによる接続はしきい値のカウントに含まれません。

TOTALDBPARTITIONCONNECTIONS のしきい値を設定する際は、TOTALSCPARTITIONCONNECTIONS に指定したしきい値に対応できる大きさの値を設定してください。例えば、データベースに 5 つのサービス・スーパークラスを定義していて、それぞれのサービス・スーパークラスで TOTALSCPARTITIONCONNECTIONS しきい値を 10 に設定している場合は、TOTALDBPARTITIONCONNECTIONS しきい値を最低でも 50 以上に設定してください。

CONCURRENTWORKLOADOCCURRENCES しきい値

CONCURRENTWORKLOADOCCURRENCES しきい値は、コーディネーター・パーティション上で同時に実行できるワークロード・オカレンスの最大数を指定する集約しきい値です。

タイプ 集約

定義ドメイン
ワークロード

適用範囲
データベース・パーティション

トラッキングされる作業
ワークロード・オカレンス

キューイング
不可

単位 並行ワークロード・オカレンスの数

予測的か反応的か
予測的

ワークロード・オカレンスの開始時に、それが生成する作業が非コーディネーター・パーティションに送信された場合、それらのパーティション上の作業は、コーディネーター・パーティション上の並行性しきい値の合計に対してカウントされません。例えば、CONCURRENTWORKLOADOCCURRENCES しきい値が、あるデータベース・パーティション上のワークロード A の 1 つのオカレンスのみを許可するように定義されていると仮定します。さらに、あるアプリケーションがデータベース・パーティション 1 に接続し、その結果ワークロード A のオカレンスが開始され、このワークロードが原因で作業がデータベース・パーティション 1、2、および 3 に送信されると仮定します。この状況では、ワークロード A のオカレンスの合計数は、データベース・パーティション 1 では 1、データベース・パーティション 2 および 3 では 0 です。そのため、別のアプリケーションがデータベース・パーティション 1 に接続し、データベース・パーティション 1 でワークロード A の別のオカレンスが開始されると、そのワークロードは拒否されます。ただしワークロード A の新規オカレンスについては、データベース・パーティション 2 および 3 で開始できます。

CONCURRENTWORKLOADACTIVITIES しきい値

CONCURRENTWORKLOADACTIVITIES しきい値は、1 つのワークロード・オカレンス内で同時に実行できるコーディネーター・アクティビティーおよびネストされたアクティビティーの最大数を指定します。

タイプ 集約

定義ドメイン
ワークロード

適用範囲
ワークロード・オカレンス

トラッキングされる作業
認識されているコーディネーター・アクティビティーおよびネストされたアクティビティー (21 ページの『アクティビティー』を参照)

キューイング
不可

単位 並行ワークロード・アクティビティーの数

予測的か反応的か
予測的

このしきい値は、単一のワークロード・オカレンスに適用されます。同時に実行されるワークロードのオカレンスが複数存在する場合、しきい値は、各ワークロード・オカレンスに別個に適用されます。トラッキングされるアクティビティーには、認識されているコーディネーター・アクティビティーすべてと、コーディネーター・アクティビティーの実行の結果として生成されたすべてのネストされたアクティビティーが含まれます。例えば、あるストアード・プロシージャーが呼び出されてそのストアード・プロシージャーが何らかの SQL を実行した場合は、CALL ステートメント (コーディネーター・アクティビティー) と、ストアード・プロシージャーが実行した SQL ステートメント (ネストされたアクティビティー) の両方が、しきい値のカウントに含められます。

COMMIT、ROLLBACK、および ROLLBACK to SAVEPOINT ステートメントは、このしきい値の影響を受けません。

ネストされたアクティビティーに関する考慮事項

このしきい値によってトラッキングされるネストされたアクティビティーは、以下の基準を満たしている必要があります。

- 認識されているコーディネーター・アクティビティーであること。 94 ページの『ワーク・クラスの作業タイプおよび SQL ステートメント』に記述されている認識済みのタイプのものではない、ネストされたコーディネーター・アクティビティーは、カウントされません。同様に、RPC 要求、DSS 要求、およびネストされた DSS 要求などのネストされたサブエージェント・アクティビティーもカウントされません。
- SQL を発行するユーザー作成のストアード・プロシージャーなどのユーザー・ロジックや、SYSPROC.ADMIN_CMD ストアード・プロシージャーから直接呼び出されること。DB2 ユーティリティーの呼び出しや、SYSIBM、SYSFUN、または SYSPROC スキーマ内の他の何らかのコードの呼び出しによって開始された、ネストされたコーディネーター・アクティビティーは、このしきい値で指定された上限へのカウントに含まれません。

例

この例では、CONCURRENTWORKLOADACTIVITIES しきい値の最大値が 5 に設定されており、ユーザー・ロジックによってワークロード・オカレンスで以下の一連の操作が行われます。

1. load コマンドを発行する: 現在のワークロード・アクティビティーの数は 1 です。
   - load コマンドが内部で何らかの SQL を発行する: 現在のワークロード・アクティビティーの数は 1 です (ユーティリティーによって生成された SQL は、CONCURRENTWORKLOADACTIVITIES しきい値のカウントに含まれません)。
   - load コマンドが終了する: 現在のワークロード・アクティビティーの数は 0 です。

2. SYSPROC.SP1 ストアード・プロシージャーを呼び出す: 現在のワークロード・アクティビティーの数は 1 です。
   - SYSPROC.SP1 ストアード・プロシージャーが何らかの SQL を生成する: 現在のワークロード・アクティビティーの数は 1 です (ユーティリティーによって生成された SQL は、CONCURRENTWORKLOADACTIVITIES しきい値のカウントに含まれません)。
   - SYSPROC.SP1 ストアード・プロシージャーが終了する: 現在のワークロード・アクティビティーの数は 0 です。
3. カーソル C1 を開く: 現在のワークロード・アクティビティーの数は 1 です。
4. runstats コマンドを発行する: 現在のワークロード・アクティビティーの数は 1 です。
   - runstats コマンドが何らかの SQL を生成する: 現在のワークロード・アクティビティーの数は 1 です。
   - runstats コマンドが終了する: 現在のワークロード・アクティビティーの数は 1 です。
5. カーソル C1 を閉じる: 現在のワークロード・アクティビティーの数は 0 です。
6. BOB.SP1 ストアード・プロシージャーを呼び出す: 現在のワークロード・アクティビティーの数は 1 です。
   - BOB.SP1 ストアード・プロシージャーが 3 つのカーソルを開く: 現在のワークロード・アクティビティーの数は 4 です。
   - BOB.SP1 ストアード・プロシージャーが SYSPROC.SP2 ストアード・プロシージャーを呼び出す: 現在のワークロード・アクティビティーの数は 5 です。
     - SYSPROC.SP2 ストアード・プロシージャーが何らかの SQL を発行する: 現在のワークロード・アクティビティーの数は 5 です。
     - SYSPROC.SP2 ストアード・プロシージャーが終了する: 現在のワークロード・アクティビティーの数は 4 です。
   - BOB.SP1 ストアード・プロシージャーが BOB.SP2 ストアード・プロシージャーを呼び出す: 現在のワークロード・アクティビティーの数は 5 です。
     - BOB.SP2 ストアード・プロシージャーが何らかの SQL を発行する: この時点で、しきい値がトリガーされます。
     - BOB.SP2 ストアード・プロシージャーが終了する: 現在のワークロード・アクティビティーの数は 4 です。
   - BOB.SP1 ストアード・プロシージャーが終了する: 現在のワークロード・アクティビティーの数は 0 です。
7. カーソル C2 を開く: 現在のワークロード・アクティビティーの数は 1 です。
8. BOB.SP2 ストアード・プロシージャーを呼び出す: 現在のワークロード・アクティビティーの数は 2 です。

CONCURRENTDBCOORDACTIVITIES しきい値

CONCURRENTDBCOORDACTIVITIES しきい値は、指定された定義ドメイン内のすべてのデータベース・パーティション全体で同時に実行できる、認識されているコーディネーター・アクティビティーの最大数を指定します。アプリケーションが複

数の並行アクティビティーを開始する場合、アプリケーションはこのしきい値を複数回渡さなければならないことがあり、場合によってはこのしきい値に関して使用可能な並行性を使い切ってセルフ・デッドロック のシナリオを引き起こす可能性があります。

タイプ 集約

定義ドメイン
データベース、ワーク・アクション、サービス・スーパークラス、サービス・サブクラス

適用範囲
データベース

トラッキングされる作業
認識されているコーディネーター・アクティビティーおよびネストされたアクティビティー (94 ページの『ワーク・クラスの作業タイプおよび SQL ステートメント』を参照)

キューイング
可

単位 並行データベース・アクティビティーの数

予測的か反応的か
予測的

このしきい値は、CONCURRENTWORKLOADACTIVITIES しきい値の汎用化です。CONCURRENTWORKLOADACTIVITIES はワークロード・ドメインで実行されるアクティビティーにのみ適用されますが、CONCURRENTDBCOORDACTIVITIES しきい値は、データベース全体から単一のワーク・アクションに及ぶさまざまなドメインに適用することができます。CONCURRENTWORKLOADACTIVITIES しきい値と同様、CONCURRENTDBCOORDACTIVITIES しきい値は、コーディネーター・アクティビティーと、生成されるすべてのネストされたアクティビティーをトラッキングします。 CONCURRENTWORKLOADACTIVITIES しきい値と異なり、CONCURRENTDBCOORDACTIVITIES しきい値はキューイングしきい値です。

CONCURRENTDBCOORDACTIVITIES タイプのキューイングしきい値を作成するときは、キューによる競合がどのような構成で発生するかを認識しておく必要があります。以下に例を示します。

1. タイプ CONCURRENTDBCOORDACTIVITIES の並行性しきい値が作成されます。最大並行値は 1、キュー・サイズは 2 以上です。
2. アプリケーションは、DB2 データ・サーバーがアクティビティー A1 と認識するカーソルを開き (あるいはストアード・プロシージャーを呼び出し) ます。これによって、しきい値に使用できるユニークなチケットが消費されます。
3. アクティビティー A1 がアクティブである間に、アプリケーションは 2 番目の SQL ステートメントを発行します。これをデータ・サーバーはアクティビティー A2 と認識します。このアクティビティーも、並行性しきい値の影響を受けます。 A1 アクティビティーが既に実行中であるため、新規のアクティビティー A2 はキューに入れられます。

この状況は、アプリケーションでは解決不可能です。アプリケーションは A2 が実行するまで待機しますが、A2 は A1 の実行が終了するまで待機しています。この状況は、外部からの介入でしか解決しません。

この例は、複数のアプリケーションとキューの場合にも一般化して考えられます。この状況は、並行値を大きくするか、または並行値が適正に設定されている場合は一部のアクティビティーをキャンセルすることによって解決します。また、時間しきい値を使用して、アクティビティーがキューの中にいつまでも残るのを防ぎます。このようにすると、上記のような場面で、外部からの介入なしに解決することができます。

しきい値の評価順序

特定のしきい値は他のしきい値から独立して評価されますが (それはこれらのしきい値が、新規接続または新規ワークロード・オカレンスなど、特定のイベントによって実行されるため)、その他のしきい値は相互に依存しており、それらを同じデータベース内で定義する場合は特定の順序で評価する必要があります。

しきい値の評価は次のように行われます。

- CONCURRENTWORKLOADOCCURRENCES。このしきい値は、このしきい値の適用対象であるワークロード定義で新規ワークロード・オカレンスが開始したときに、独立して評価されます。
- TOTALDBPARTITIONCONNECTIONS。このしきい値は、データベースに新規の接続が行われると、独立して評価されます。
- TOTALSCPARTITIONCONNECTIONS。このしきい値は、接続がサービス・クラスに結合する時 (新規の接続、またはワークロードの再割り当ての結果としてのサービス・クラス間の転送のどちらか) に、評価されます。

残りのしきい値はすべて、SQL ステートメントまたはロード・ユーティリティーなどのユーティリティーの実行の結果による認識されたアクティビティーに基づいています。予測的しきい値は、反応的しきい値よりも前に検査されます。それは、予測的しきい値に違反していないことを確認するための検査をしてからでなければ、データベース・アクティビティーの実行を開始できないためです。予測的しきい値が評価される順序は以下のとおりです。

注: しきい値を定義していない場合、このステップは省略されます。パフォーマンス上の理由で、実行時に、説明されているステップがまとめられる可能性もあります。

1. ESTIMATEDSQLCOST しきい値があるかどうかを検査し、ある場合はそれに違反していないかどうかを検査します。このしきい値を複数のドメイン内に定義すると、しきい値は有効範囲の解決の規則に従って解決されます (詳細については、70 ページの『アクティビティーしきい値の有効範囲の解決』を参照してください)。この操作の結果は、そのアクティビティーに適用可能な ESTIMATEDSQLCOST の 1 つの値です。
2. ESTIMATEDSQLCOST しきい値があるかどうかを検査し、ある場合はそれに違反していないかどうかを検査します。しきい値に違反している場合は、対応するアクションが実行されます。当てはまる場合は、次のステップへ進みます。

3. CONCURRENTWORKLOADACTIVITIES しきい値があるかどうかを検査し、ある場合はそれに違反していないかどうかを検査します。しきい値に違反している場合は、対応するアクションが実行されます。当てはまる場合は、次のステップへ進みます。
4. CONCURRENTDBCOORDACTIVITIES しきい値があるかどうかを検査し、ある場合はそれに違反していないかどうかを検査します。しきい値に違反している場合は、対応するアクションが実行されます。当てはまる場合は、次のステップへ進みます。
5. CONCURRENTDBCOORDACTIVITIES しきい値があるかどうかを検査し、ある場合はそれに違反していないかどうかを検査します。しきい値に違反している場合は、対応するアクションが実行されます。当てはまる場合は、次のステップへ進みます。
6. CONCURRENTDBCOORDACTIVITIES しきい値があるかどうかを検査し、ある場合はそれに違反していないかどうかを検査します。しきい値に違反している場合は、対応するアクションが実行されます。当てはまる場合は、次のステップへ進みます。
7. CONCURRENTDBCOORDACTIVITIES しきい値があるかどうかを検査し、ある場合はそれに違反していないかどうかを検査します。しきい値に違反している場合は、対応するアクションが実行されます。当てはまる場合は、次のステップへ進みます。

並行性しきい値の評価順序は、アクティビティーしきい値の解決に使用された階層に従いません (詳細については、70 ページの『アクティビティーしきい値の有効範囲の解決』を参照してください)。データベース・レベルのワーク・アクション・セットの並行性しきい値は、以下の状態を回避するために最初に検査されます。以下のしきい値が定義されていると仮定します。

- LOAD アクティビティーに対するワーク・アクションの並行性しきい値が 1 の値で定義されています。
- サービス・スーパークラス S1 の並行性限度が 10 に設定されています。

また、1 つの LOAD アクティビティーがデータベース内 (任意のサービス・スーパークラス下) で既に実行中で、そして 9 つのアクティビティーが既にサービス・スーパークラス S1 で実行されている時に、新規の LOAD アクティビティーが 10 番目のアクティビティーとして入ってくると想定します。有効範囲の解決の階層がしきい値の評価に使用されると、着信 LOAD アクティビティーはサービス・クラスのしきい値に違反しないことになり、並行性が 10 に増加します。次に LOAD アクティビティーはワーク・アクションのしきい値の並行性限度に照らして評価されますが、これは違反になります。データベースで LOAD アクティビティーが既に実行中であり、ワーク・アクションのしきい値の並行性の値は 1 に限られているからです。こうして 10 番目の LOAD アクティビティーはキューに入れられます。

この状態の結果として、サービス・スーパークラス S1 に到着する新規アクティビティーはどれもキューに入れられます (サービス・クラスの並行性限度に既に達したため)。ワーク・アクションのしきい値のキューはサービス・クラスに影響を与えますが、サービス・クラス内で実行しようとしているアクティビティーは必ずしもワーク・アクションのしきい値の条件と関係するわけではないので (例えば、サービス・スーパークラス S1 で実行しようとしている挿入操作は、ワーク・アクションのしきい値の条件のためにキューに入れられた LOAD アクティビティーを待つ必

要はありません)、それは望ましいことではありません。それで、こうした種類の状態を回避するためにワーク・アクションの並行性しきい値が最初に検査されます。並行性しきい値が最初に検査されるので、サービス・クラス内の 10 番目のアクティビティー (ここでは LOAD アクティビティー) が、サービス・スーパークラス S1 内の 1 つの場所を占めてしまう前に、ワーク・アクションのしきい値レベルでブロックされます。

反応的しきい値に関する考慮事項

反応的しきい値はアクティビティーの実行中に離散的な方式で評価されます。反応的しきい値 SQLTEMPSPACE、SQLROWSRETURNED、または ACTIVITYTOTALTIME を評価するのに特定の順序は使用されません。

しきい値の作業

しきい値の作成

DDL ステートメント CREATE THRESHOLD を使用してしきい値を作成します。しきい値は、リソースの使用量に制限を設けるために作成します。

しきい値を作成するためには、DBADM または SYSADM 権限が必要です。

前提条件について詳しくは、以下のトピックを参照してください。

- 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』
- 命名規則

ワーク・アクション・セットのしきい値を作成するには、CREATE WORK ACTION SET ステートメントまたは ALTER WORK ACTION SET ステートメントを ADD WORK ACTION キーワードとともに使用します。詳しくは、CREATE WORK ACTION SET ステートメントまたは ALTER WORK ACTION SET ステートメントを参照してください。

しきい値を作成するには、次のようにします。

1. CREATE THRESHOLD ステートメントを発行し、以下に挙げるしきい値のプロパティーを 1 つ以上指定します。
   - しきい値の名前。
   - しきい値のドメイン。しきい値のドメインは、しきい値が付加されて、しきい値が作用するデータベース・オブジェクトです。

     注: 適用されるドメインは、しきい値のタイプに依存します。詳しくは、65 ページの『しきい値のサマリー』を参照してください。
     ドメインとして使用できるものには以下があります。
     - データベース
     - 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

     詳しくは、63 ページの『第 4 章 しきい値』を参照してください。

- しきい値の適用範囲。しきい値の有効範囲は、ドメイン内でのしきい値の強制範囲です。

  注: 適用される適用範囲は、しきい値のタイプに依存します。詳しくは、65 ページの『しきい値のサマリー』を参照してください。
  適用範囲は、以下のいずれかになります。

  – データベース
  – データベース・パーティション
  – ワークロード・オカレンス

  詳しくは、63 ページの『第 4 章 しきい値』を参照してください。
- オプション: しきい値が使用可能か使用不可かを指定する値。デフォルトでは、しきい値は使用可能として作成されます。しきい値を使用不可として作成し、後で使用可能にする場合は、ALTER THRESHOLD ステートメントを使用します。詳しくは、84 ページの『しきい値の変更』を参照してください。

  注: ワーク・アクションしきい値がある場合は、ALTER WORK ACTION SET ステートメントを使用して、しきい値を使用可能または使用不可にします。
- しきい値の最大値を指定するしきい値述部。最大値に違反すると、しきい値に指定されたアクションが強制的に施行されます。しきい値は以下のとおりです。

  – TOTALDBPARTITIONCONNECTIONS。 0 または正整数を指定して、データベース・パーティションで実行できる並行コーディネーター接続の最大数を指定します。値 0 を指定した場合は、新規のコーディネーター接続を接続させません。定義ドメインは DATABASE に、適用範囲は PARTITION にする必要があります。例えば、データベース・パーティション上で実行できる並行コーディネーター接続の数を 5 つにする場合は、次のようにします。

    ```
    TOTALDBPARTITIONCONNECTIONS > 5
    ```

    注: このしきい値は、DBADM 権限を持つユーザーには適用されません。
  – TOTALSCPARTITIONCONNECTIONS。 0 または正整数を指定して、サービス・スーパークラスにおいてデータベース・パーティション上で実行できる並行コーディネーター接続の最大数を指定します。値 0 を指定した場合は、新規の接続をサービス・スーパークラスに加わらせません。定義ドメインは SERVICE SUPERCLASS に、適用範囲は PARTITION にする必要があります。このしきい値はキューイングしきい値であるため、オプションの AND QUEUEDCONNECTIONS キーワードをとります。有効な引数は以下のとおりです。

    - AND QUEUEDCONNECTIONS > 0 は、コーディネーター接続をキューに入れないことを示します。
    - AND QUEUEDCONNECTIONS > *integer* は、コーディネーター接続が最大数を超えた場合の最大キュー・サイズを示します。
    - AND QUEUEDCONNECTIONS UNBOUNDED は、キュー・サイズに上限がないことを示します。この場合、しきい値の違反条件が満たされることはありません。

例えば、あるサービス・スーパークラスにおいて、データベース・パーティション上で実行できる並行コーディネーター接続の数を 5 つ、キューに入れることのできる接続の数を 7 つとする場合は、次のようにします。

```
TOTALSCPARTITIONCONNECTSIONS > 5 AND QUEUEDCONNECTIONS > 7
```

注: TOTALSCPARTITIONCONNECTIONS を使用して、手動では使用不可にできないサービス・クラス (例えば、デフォルト・ユーザー・クラスなど) を効果的に使用不可にすることができます。SYSDEFAULTADMWORKLOAD サービス・クラスで実行される DBADM 権限を持つユーザーにはしきい値が適用されませんが、使用不可にされたサービス・クラスはどのユーザーにも使用できなくなります。

– CONNECTIONIDLETIME。正整数と以下のいずれかのキーワードを指定して、接続をアイドル状態のままにしておくことができる最大時間を指定します。

  - DAY
  - DAYS
  - HOUR
  - HOURS
  - MINUTE
  - MINUTES

  最小細分が 5 MINUTES のため、指定される値はすべて、最も近いゼロ以外の 5 MINUTESの倍数に丸められます。定義ドメインは DATABASE または SERVICE SUPERCLASS に、適用範囲は DATABASE にする必要があります。例えば、接続をアイドルにしておける時間を 90 分にする場合は、次のように指定します。

  ```
  CONNECTIONIDLETIME > 90 MINUTES
  ```

  このしきい値に指定できる値の最大値は 2147483400 秒です。2147483400 秒よりも大きな値が (DAY、HOUR、MINUTE、または SECOND キーワードを使用して) 指定された場合は、すべて最大値で切り捨てられます。

– CONCURRENTWORKLOADOCCURRENCES。ゼロ以外の正整数を指定して、データベース・パーティション上の並行ワークロード・オカレンスの最大数を指定します。定義ドメインは WORKLOAD にする必要があります。例えば、データベース・パーティション上に同時に存在できるワークロード・オカレンスの最大数を 8 にする場合は、次のようにします。

  ```
  CONCURRENTWORKLOADOCCURRENCES > 8
  ```

– CONCURRENTWORKLOADACTIVITIES。ゼロ以外の正整数を指定して、ワークロード・オカレンスにおけるデータベース・パーティション上の並行コーディネーター・アクティビティーおよびネストされたアクティビティーの最大数を指定します。適用範囲は WORKLOAD OCCURRENCE にする必要があります。例えば、あるワークロード・オカレンスにおいてデータベース・パーティション上で並行して実行できるアクティビティーの最大数を 26 にする場合は、次のようにします。

  ```
  CONCURRENTWORKLOADACTIVITIES > 26
  ```

– CONCURRENTDBCOORDACTIVITIES。 0 または正整数を指定して、指定されたドメイン内の全データベース・パーティションで実行できる並行コ

ーディネーター・アクティビティーの最大数を指定します。値 0 を指定した場合は、新規のコーディネーター・アクティビティー (このしきい値を使用不可にしたり変更したりするためのアクティビティーもすべて含む) を実行させません。この場合、しきい値を使用不可にしたり変更したりするには SYSDEFAULTADMWORKLOAD ワークロードを使用する必要があります。定義ドメインは、DATABASE、SERVICE SUPERCLASS、または SERVICE SUBCLASS にすることができます。適用範囲は DATABASE にする必要があります。このしきい値はキューイングしきい値であるため、オプションの AND QUEUEDACTIVITIES キーワードをとります。有効な引数は以下のとおりです。

- AND QUEUEDACTIVITIES > 0 は、コーディネーター・アクティビティーをキューに入れないことを示します。
- AND QUEUEDACTIVITIES > *integer* は、コーディネーター・アクティビティーが最大数を超えた場合の最大キュー・サイズを示します。
- AND QUEUEDACTIVITIES UNBOUNDED は、キューのサイズに上限がないことを示します。この場合、しきい値の違反条件が満たされることはありません。

例えば、指定した定義ドメインで実行できる並行コーディネーター・アクティビティーの最大数を 12 に、キューに入れることのできるアクティビティーの数を 9 にする場合は、次のようにします。

```
CONCURRENTDBCOORDACTIVITIES > 12 AND QUEUEDACTIVITIES > 9
```

– ESTIMATEDSQLCOST。ゼロ以外の正の 64 ビット整数を指定して、オプティマイザーが割り当てる、コーディネーター DML アクティビティーまたはユーザー・ロジックによって呼び出される DML アクティビティーのコストの最大値を指定します。定義ドメインは、DATABASE、SERVICE SUPERCLASS、または SERVICE SUBCLASS にすることができます。適用範囲は DATABASE にする必要があります。例えば、1,000 timeron を超える大きさの DML アクティビティーをデータベース内で実行しないようにする場合は、次のように指定します。

```
ESTIMATEDSQLCOST > 1000
```

– SQLROWSRETURNED。ゼロ以外の正整数を指定して、アプリケーション・サーバーからクライアント・アプリケーションに返すことができる行の最大数を指定します。定義ドメインは、DATABASE、SERVICE SUPERCLASS、または SERVICE SUBCLASS にすることができます。適用範囲は DATABASE にする必要があります。例えば、返せる行数を 50,000 行までにする場合は、次のように指定します。

```
SQLROWSRETURNED > 50000
```

– ACTIVITYTOTALTIME。ゼロ以外の正整数を指定して、アクティビティーの実行時間の上限を指定します。この実行時間には、アクティビティーがキューに入っている可能性のある時間も含まれます。この値の後に、以下のいずれかの期間キーワードを指定します。

- DAY
- DAYS
- HOUR
- HOURS

- MINUTE
- MINUTES

最小細分が 5 MINUTES のため、指定される値はすべて、最も近いゼロ以外の 5 MINUTES の倍数に丸められます。定義ドメインは、DATABASE、SERVICE SUPERCLASS、または SERVICE SUBCLASS にすることができます。適用範囲は DATABASE にする必要があります。例えば、アクティビティー時間の上限を 2 時間にする場合は、次のように指定します。

```
ACTIVITYTOTALTIME > 2 HOURS
```

このしきい値に指定できる値の最大値は 2147483400 秒です。2147483400 秒よりも大きな値が (DAY、HOUR、MINUTE、または SECOND キーワードを使用して) 指定された場合は、すべて最大値で切り捨てられます。

– SQLTEMPSPACE。ゼロ以外の正整数を指定して、データベース・パーティションにおいて消費できる TEMPORARY 表スペースの最大量を指定します。この値の後に、以下のいずれかのスペース・キーワードを指定します。

- K (キロバイト)
- M (メガバイト)
- G (ギガバイト)

定義ドメインは、DATABASE、SERVICE SUPERCLASS、または SERVICE SUBCLASS にすることができます。適用範囲は DATABASE PARTITION にする必要があります。例えば、データベース・パーティションで使用できる TEMPORARY 表スペースの最大量を 100 MB に指定する場合は、次のようにします。

```
SQLTEMPSPACE > 100 M
```

• しきい値の境界に違反したときに実行するアクション。アクションには、必須の進行アクションと、オプションのアクティビティー収集アクションがあります。アクティビティーの収集アクションでは、しきい値の境界に違反する原因となったアクティビティーについて、どの情報を収集するかを指定します。アクションは、アクティビティー関連のしきい値についてのみ指定します。アクティビティー関連ではないしきい値にアクションが指定された場合、アクションは無視されます。

– STOP EXECUTION。TOTALDBPARTITIONCONNECTIONS しきい値や TOTALSCPARTITIONCONNECTIONS しきい値の場合は、接続が確立されなくなります。CONNECTIONIDLETIME しきい値の場合は、接続が閉じられます。CONCURRENTWORKLOADOCCURRENCES の場合は、新規のワークロード・オカレンスが作成されなくなります。他のすべてのアクティビティー関連のしきい値の場合は、しきい値の違反の原因となったアクティビティーが停止します。THRESHOLDVIOLATIONS イベント・モニターがアクティブな場合は、しきい値に違反したことを示すレコードがイベント・モニターに書き込まれます。

– CONTINUE。THRESHOLDVIOLATIONS イベント・モニターがアクティブな場合は、しきい値に違反したことを示すレコードがイベント・モニターに書き込まれます。イベント・モニターが存在しない場合は、それ以上のアクションは行われません。

注: キューイングしきい値に CONTINUE がしきい値アクションとして指定されている場合は、キュー・サイズにどんな厳しい値が指定されていても関係なく、キューのサイズが事実上無制限になります。

– COLLECT ACTIVITY DATA。アクティビティー・イベント・モニター用に収集する情報を指定します。デフォルトは COLLECT ACTIVITY DATA NONE です。

  - NONE。しきい値に違反したアクティビティーが実行を完了する際に、アクティビティー情報を収集しません。
  - データを収集するロケーション:
    - ON COORDINATOR DATABASE PARTITION。ACTIVITIES イベント・モニターがアクティビティーのデータベース・パーティションでアクティブである場合、そのコーディネーター・パーティションでのみ、アクティビティー・データが収集されます。
    - ON ALL DATABASE PARTITIONS。アクションが CONTINUE になっている予測的しきい値についてのみ、そのアクティビティーが処理されるすべてのデータベース・パーティションでアクティビティー・データを収集します。ただし、アクティビティーの詳細や値は、コーディネーター・パーティションでのみ収集されます。反応的しきい値やアクションが STOP EXECUTING になっているしきい値の場合、このオプションは ON COORDINATOR DATABASE PARTITION と同じ効果を持ちます。アクティビティー・データは、ACTIVITIES イベント・モニターがアクティブであるデータベース・パーティションでのみ収集されます。
  - WITHOUT DETAILS。しきい値に違反する各アクティビティーについての情報が、アクティビティーの実行完了時に該当するイベント・モニターに送信されます。ただし、ステートメントやコンパイル環境の情報はイベント・モニターに送信されません。
  - WITH DETAILS。ステートメントやコンパイル環境の情報があるアクティビティーについて、それらの情報が該当するイベント・モニターに送信されます。詳細を要求する場合は AND VALUES を指定し、入力データ値を持つアクティビティーについて入力データ値の情報を該当するイベント・モニターに送信することもできます。

  例えば、しきい値条件に違反したアクティビティーについて、入手可能なすべての情報 (しきい値違反の原因となったステートメント、コンパイル環境、および入力データ値を含む) を、アクティビティー完了時にアクティビティー・イベント・モニターに送信する場合は、CREATE THRESHOLD ステートメントに次のキーワードを使用します。

  ```
  COLLECT ACTIVITY DATA ON ALL DATABASE PARTITIONS WITH DETAILS AND VALUES
  ```

2. 変更をコミットします。 変更をコミットすると、しきい値が SYSCAT.THRESHOLDS ビューに追加されます。

しきい値の変更

DDL ステートメント ALTER THRESHOLD を使用してしきい値を変更します。期待どおりの結果が得られないときは、しきい値に変更を加えることができます。

しきい値を変更するためには、DBADM または SYSADM 権限が必要です。

前提条件について詳しくは、 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』を参照してください。

ワーク・アクション・セットのしきい値を変更するには、ALTER WORK ACTION SET ステートメントを ADD WORK ACTION キーワードとともに使用します。詳しくは、ALTER WORK ACTION SET ステートメントを参照してください。

しきい値を変更するには、次のようにします。

1. ALTER THRESHOLD ステートメントで、以下に挙げるしきい値のプロパティーを 1 つ以上指定します。変更できるのは以下の各プロパティーです。これらのプロパティーについてサポートされる値の説明は、 79 ページの『しきい値の作成』を参照してください。
   - しきい値述部の新しい境界。

     注: しきい値のタイプは変更できません (つまり、TOTALDBPARTITIONCONNECTIONS しきい値を TOTALSCPARTITIONCONNECTIONS しきい値に変更することはできません)。
   - しきい値の境界に違反したときに実行するアクション。
   - しきい値が使用可能か使用不可か。アクティビティーは、アクティビティーの開始時に使用可能であった適用可能なしきい値の下で実行されます。
2. 変更をコミットします。 変更をコミットすると、しきい値が SYSCAT.THRESHOLDS ビューで更新されます。

しきい値のドロップ

DDL ステートメント DROP THRESHOLD を使用して、必要でなくなったしきい値をドロップします。

しきい値をドロップするためには、DBADM または SYSADM 権限が必要です。

前提条件について詳しくは、 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』を参照してください。

注: ワーク・アクション・セット内のしきい値をドロップする場合は、ALTER WORK ACTION SET ステートメントを使用してください。

しきい値をドロップするには、次のようにします。

1. しきい値がキューイングしきい値である場合は、ALTER THRESHOLD ステートメントを使用して、しきい値を使用不可にします。
2. ALTER THRESHOLD ステートメントを使用してキューイングしきい値を使用不可にしたなら、COMMIT ステートメントを発行して変更をコミットします。
3. DROP THRESHOLD ステートメントを使用して、しきい値をドロップします。
4. 変更をコミットします。 変更をコミットすると、しきい値が SYSCAT.THRESHOLDS ビューから除去されます。

第 5 章 ワーク・アクション・セット、ワーク・アクション、ワーク・クラス・セット、およびワーク・クラス

ワークロードを使用してアクティビティーをサブミット元 に基づいて分類するのと同様の方法で、ワーク・アクション・セット、ワーク・アクション、ワーク・クラス・セット、およびワーク・クラスを使用してアクティビティーをその種類 に基づいて分類できます。アクティビティー・タイプなどの属性に基づいてアクティビティーを分類する場合、これらのアクティビティーに異なったタイプのアクションを適用することによって処理方法を変えることができます。

アクティビティーをさまざまなワーク・クラスに分類でき、ワーク・クラスをワーク・クラス・セットにグループ化することができます。ワーク・アクション・セット (これにはワーク・アクションを入れることができる) をワーク・クラス・セットに適用すると、ワーク・アクション・セット内のワーク・アクションをワーク・クラス・セット内のワーク・クラスに適用できます。アクティビティーを分類するオブジェクト (ワーク・クラスおよびワーク・クラス・セット) は、アクティビティーに適用されるアクションを定義するオブジェクト (ワーク・アクションおよびワーク・アクション・セット) から分離されているので、種別オブジェクトを複数のワーク・アクション・セットおよびワーク・アクション間で共有できます。

ワーク・クラスとワーク・クラス・セット

ワーク・クラスとは、データベース・アクティビティーの属性に基づいて個々のデータベース・アクティビティーを分類する方法のことです。ワーク・クラスは、さまざまなワーク・アクション・セットが共有できるワーク・クラス・セットにグループ分けされます。

どのワーク・クラスにアクティビティーが関連付けられるかを決定できるデータベース・アクティビティーの属性の例には、アクティビティー・タイプ (DDL、DML、LOAD)、見積コスト (使用可能な場合)、見積カーディナリティー (使用可能な場合)、およびスキーマ (使用可能な場合) があります。

ワーク・クラスには以下の属性があります。

- ワーク・クラス名。これはワーク・クラス・セット内で固有でなければなりません。
- データベース・アクティビティー属性。これは以下の情報で構成されます。
  - このワーク・クラスに分類されるデータベース・アクティビティーのタイプ。

    定義済みキーワード (例えば、READ、WRITE、DML、DDL、および LOAD) を使用して、データベース要求をさまざまなカテゴリーに分類することができます。異なるタイプのデータベース・アクティビティーを、その作業タイプに基づいて、1 つのワーク・クラスに関連付けることができます。例えば、WRITE キーワードには、UPDATE、DELETE、INSERT、MERGE と、DELETE、INSERT、または UPDATE を含む SELECT が含まれます。詳しくは、94 ページの『ワーク・クラスの作業タイプおよび SQL ステートメント』を参照してください。

© Copyright IBM Corp. 2007, 2008

– データベース・アクティビティーの DML または XQuery タイプを詳細に分類する範囲情報。
  - 指定する範囲のタイプ (timeron コストまたはカーディナリティーのいずれか)。値の範囲の指定はオプションです。例えば、ワーク・クラスの範囲を指定する場合、見積コストが 100 timeron 未満のすべての照会を、他の照会とは異なる仕方で処理するように指定できます。
  - 範囲の下限。
  - 範囲の上限。
– 呼び出されるルーチンのスキーマ。

  ワーク・クラスの定義時に、呼び出されるプロシージャーのスキーマに応じて CALL ステートメントをさらに詳細に分類するためにスキーマ属性を使用できます。例えば、ワーク・クラスのスキーマに SCHEMA1 を指定し、作業タイプが CALL の場合、SCHEMA1 プロシージャーを呼び出すすべての CALL ステートメントはそのワーク・クラスに分類されます。 CALL または ALL 以外のワーク・クラス・タイプにスキーマを指定すると、エラー SQL0628N が返されます。

- ワーク・クラスの評価順序 (またはワーク・クラス・セット内のワーク・クラスの位置)。

  詳しくは、 96 ページの『ワーク・クラス・セット内のワーク・クラスの評価順序』を参照してください。
- ワーク・クラスを一意的に識別する自動生成クラス ID。

ワーク・クラスは以下の 2 とおりの方法で作成できます。

- 新規ワーク・クラスを含む新規ワーク・クラス・セットを作成するには、CREATE WORK CLASS SET ステートメントの WORK CLASS キーワードを使用します。
- 新規ワーク・クラスを既存のワーク・クラス・セットに追加するには、ALTER WORK CLASS SET ステートメントの ADD キーワードを使用します。

ワーク・クラスを変更するには、ALTER WORK CLASS SET ステートメントの ALTER WORK CLASS キーワードを使用することができます。

ワーク・クラス・セットからワーク・クラスをドロップするには、ALTER WORK CLASS SET ステートメントの DROP WORK CLASS キーワードを使用し、ワーク・クラス・セットをドロップするには、DROP WORK CLASS SET ステートメントを使用することができます。

ワーク・クラスを表示するには、SYSCAT.WORKCLASSES ビューを照会します。

1 つ以上のワーク・クラスをグループ化するには、ワーク・クラス・セットを使用します。ワーク・クラス・セットは以下の属性で構成されます。

- ワーク・クラス・セットの固有の記述名。
- ワーク・クラス・セットに提供するコメント。
- ゼロ以上のワーク・クラス (ワーク・クラスはワーク・クラス・セット内でしか存在できませんが、ワーク・クラス・セットにはワーク・クラスが含まれていなくても構いません)。

- ワーク・クラス・セットを一意的に識別する自動生成 ID。

新規ワーク・クラス・セットを作成するには、CREATE WORK CLASS SET ステートメントを使用します。空のワーク・クラス・セットを作成して後からワーク・クラスを追加することもできますし、1 つ以上のワーク・クラスを含むワーク・クラス・セットを作成することもできます。

既存のワーク・クラス・セットを変更するには、ALTER WORK CLASS SET ステートメントを使用して以下の方法で実行できます。

- ワーク・クラス・セットにワーク・クラスを追加します。
- ワーク・クラス・セット内のワーク・クラスのワーク・クラス属性を変更します。
- ワーク・クラス・セットからワーク・クラスをドロップします。

ワーク・クラス・セット属性は変更できません。

ワーク・クラス・セットをドロップするには、DROP WORK CLASS SET ステートメントを使用します。

ワーク・クラス・セットを表示するには、SYSCAT.WORKCLASSSETS カタログ・ビューを照会します。

以下の図は、ワーク・クラス・セット内のワーク・クラスの例を示しています。

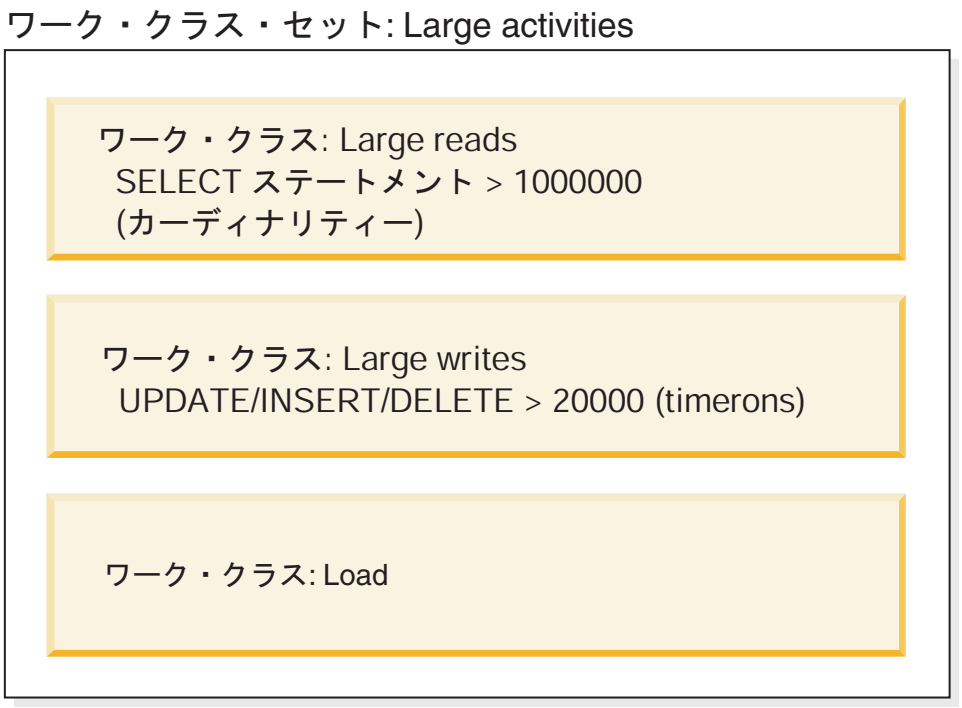


図 14. ワーク・クラスおよびワーク・クラス・セットの例

システム上でワーク・クラス・セットが有効であるには、ワーク・アクション・セットを定義して、それをワーク・クラス・セットと関連付けなければなりません。ワーク・アクション・セットを使用することで、ワーク・クラス・セットをサービス・スーパークラスまたはデータベースのどちらかと関連付けて、どのアクション

が分類内のデータベース・アクティビティーに適用されるかを示すことができます。ワーク・クラス・セットに対してワーク・アクション・セットを作成しない場合、データ・サーバーはワーク・クラス・セットを無視します。

ワーク・アクションとワーク・アクション・セット

ワーク・アクションは、ワーク・クラスと併用される場合、特定のタイプのアクティビティーを制御するために使用できます。例えば、さまざまなワーク・アクションを LOAD アクティビティーに適用して、それらが DML とは異なる仕方で処理されるようにすることができます。ワーク・アクションはワーク・アクション・セットにグループ化されます。

ワーク・アクションは以下の属性で構成されます。

- ユーザー提供のワーク・アクション名。これはワーク・アクション・セット内で固有でなければなりません。
- ワーク・アクションが適用されるワーク・クラス ID。

  ワーク・クラスに適用されるワーク・アクションがない場合、データ・サーバーはワーク・クラスが存在しないかのように動作します。ワーク・クラスに対しては複数のワーク・アクションを定義できますが、それぞれのワーク・アクションは、そのワーク・クラスに対して異なるアクションを実行するものでなければなりません。
- ワーク・クラスに一致するデータベース・アクティビティーに適用されるアクション。ワーク・アクションの有効なアクションのタイプは、ワーク・アクションが属するワーク・アクション・セットが、データベースに適用されるか、それともサービス・スーパークラスに適用されるかに応じて異なります。

  ワーク・アクション・セットが (ワーク・アクションが関連付けられるワーク・クラスに応じて) データベースに適用される場合、ワーク・アクション・セットはデータベース・アクティビティーとなる一部またはすべてのアクティビティーに適用されます。ワーク・アクション・セットが (ワーク・アクションが関連付けられるワーク・クラスに応じて) サービス・スーパークラスに適用される場合、ワーク・アクション・セットはサービス・スーパークラスの下で実行する一部またはすべてのアクティビティーに適用されます。以下に例を示します。
  - データベースに適用されるワーク・アクション・セットには、しきい値ワーク・アクションを含めることができます。しきい値ワーク・アクションが定義されているワーク・クラスにアクティビティーが割り当てられた場合、しきい値はそのアクティビティーに適用されます。
  - サービス・スーパークラスに適用されるワーク・アクション・セットには、アクティビティーをサービス・スーパークラス内のサービス・サブクラスにマップするワーク・アクションを含めることができます。アクティビティーがワーク・クラス・セット内の特定のワーク・クラスに対応し、ワーク・アクション・セットがそのワーク・クラスに定義されたマッピング・ワーク・アクションを持つ場合、そのアクティビティーは、ワーク・アクションにより指定されたサービス・サブクラスにマップされます。

  サポートされるアクションのリストについては、 97 ページの『ワーク・アクションとワーク・アクション・セットのドメイン』を参照してください。

- 指定されたアクションのターゲットであるオブジェクト。

  アクションに応じて、オブジェクトは、アクティビティーのマップ先のサービス・サブクラスとすることもできますし、どのしきい値をアクティビティーに適用するかを指定するしきい値とすることもできます。あるいは、アクションが実行を妨げるもの、収集アクションの 1 つ、あるいはカウント・アクティビティーである場合には、NULL にすることもできます。
- 特定のインターバル中、このワーク・アクションの割り当て先のワーク・クラスに関連するアクティビティーがどれだけの時間に渡って稼働しなければならなかったか (マイクロ秒数) についての統計情報を収集するためのヒストグラムを記述するテンプレート。

  この情報は、ワーク・アクション・タイプが COLLECT AGGREGATE ACTIVITY DATA (BASE または EXTENDED のいずれか) の場合にのみ収集されます。ヒストグラムおよびヒストグラム・テンプレートについて詳しくは、140 ページの『ワークロード管理のヒストグラム』を参照してください。
- ワーク・アクションが使用可能かどうか。
- ワーク・アクションを識別する自動生成 ID。

ワーク・アクションを作成するには、CREATE WORK ACTION SET ステートメントの WORK ACTION キーワード、または ALTER WORK ACTION SET ステートメントの ADD キーワードのいずれかを使用できます。ワーク・アクションを変更するには、ALTER WORK ACTION SET ステートメントの ALTER キーワードを使用できます。ワーク・アクションをワーク・アクション・セットから除去するには、ALTER WORK ACTION SET ステートメントの DROP キーワードを使用できます。

ワーク・アクションを表示するには、SYSCAT.WORKACTIONS ビューを照会します。

ワーク・アクション・セットは以下の属性で構成されます。

- データベース内で固有のワーク・アクション・セット名。
- アクションのグループが適用されるワーク・クラスを含むワーク・クラス・セットの名前。

  ワーク・クラス・セットの定義は、それらに定義されているワーク・アクション・セットからは分離されているため、ワーク・クラス・セットに対して複数のワーク・アクション・セットを定義することができます。
- ワーク・アクション・セットが関連付けられるオブジェクト (データベースまたはサービス・スーパークラス)。
- アクションとワーク・クラス・セットが適用されるサービス・スーパークラスの名前 (サービス・スーパークラスと関連付けられたワーク・アクション・セットの場合)。
- ワーク・アクション・セットが使用可能かどうか。
- ユーザー・コメント。
- 1 つ以上のワーク・アクション (ワーク・アクション・セットには必ずしもワーク・アクションが含まれている必要はありません)。

- ワーク・アクション・セットを一意的に識別する自動生成 ID。

ワーク・アクション・セットを作成するには CREATE WORK ACTION SET ステートメントを、ワーク・アクション・セットを変更するには ALTER WORK ACTION SET ステートメントを、およびワーク・アクション・セットをドロップするには DROP WORK ACTION SET ステートメントを使用することができます。

ワーク・アクション・セットを表示するには、SYSCAT.WORKACTIONSETS ビューを照会します。

ワーク・アクション・セットを作成する場合は、ワーク・アクション・セットが適用されるオブジェクトを指定する必要があります。有効なオブジェクト・タイプは、データベースまたはサービス・スーパークラスです。ワーク・アクション・セットと共に機能するワーク・クラス・セットも指定する必要があります。これによりワーク・クラス・セット内のワーク・クラスを使用して、ワーク・アクションを適用するアクティビティーのタイプを識別することができます。

そのデータベース・アクティビティーがサービス・サブクラスに直接マップするようにワークロードがセットアップされている場合、そのサービス・スーパークラスと関連付けられたワーク・アクション・セットは、そのワークロードにより実行されるアクティビティーに使用されることはありません。言い換えれば、ワークロードがアクティビティーをサービス・サブクラスに直接マップしている場合、ワーク・アクション・セットはバイパスされます。ワーク・アクション・セット内のどのワーク・アクションも、サービス・サブクラスに直接マップされているアクティビティーには適用されません。

どのようにワーク・クラス、ワーク・クラス・セット、ワーク・アクション、およびワーク・アクション・セットが一緒に動作し、他の DB2 オブジェクトに関連付けられるか

ワーク・クラスおよびワーク・アクションは一緒に動作し、特定のアクションを特定のアクティビティー・タイプに適用します。これがどのように動作するか、例を使って説明します。

以下の図は、ワーク・クラス、ワーク・クラス・セット、ワーク・アクション、およびワーク・アクション・セットがどのように一緒に動作し、他の DB2 オブジェクトと関連付けられるかについての概要を示しています。

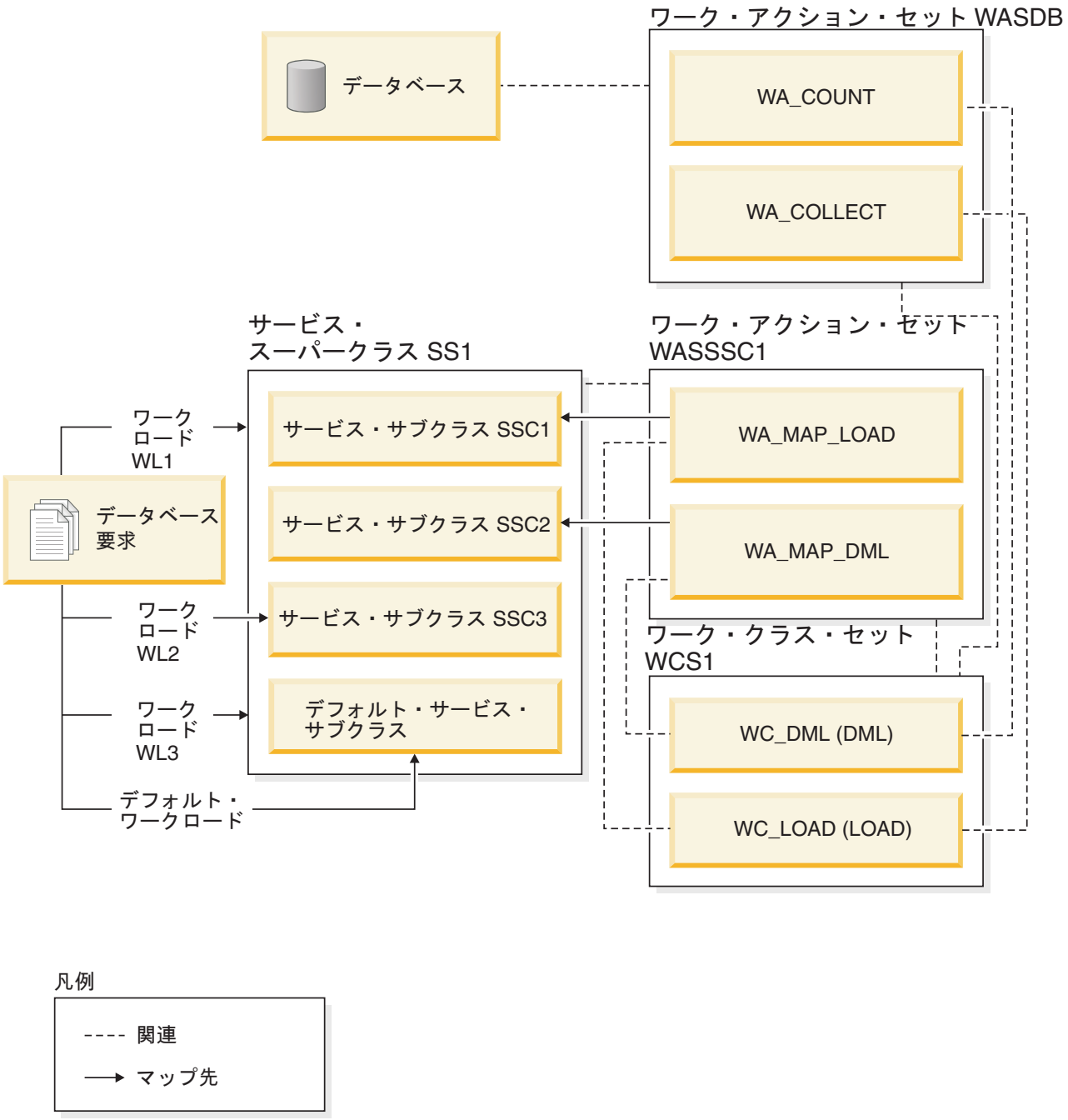


図 15. ワーク・アクション・セットとワーク・クラス・セットの概要

この図では、一部のデータベース・アクティビティーは、ワークロード WL1、ワークロード WL3、およびデフォルトのユーザー・ワークロードである SYSDEFAULTUSERWORKLOAD から、サービス・スーパークラス SS1 にマップされています。ワーク・アクション・セット WASDB はデータベースに適用されているので、デフォルトのユーザー・ワークロード、WL1 ワークロード、または WL3 ワークロードに割り当てられているアクティビティー、および WC_DML または WC_LOAD ワーク・クラスに分類されるアクティビティーは、それに適用される WASDB ワーク・アクション・セット内のワーク・アクションを持ちます。つまり、DML 作業タイプのアクティビティーはカウントされ、LOAD 作業タイプの

アクティビティーはそれに関するアクティビティー・データが収集されて、アクティブ・イベント・モニターに書き込まれます (使用可能な場合)。

ワーク・アクション・セット WASSSC1 はサービス・スーパークラス SS1 に適用されます。デフォルトのユーザー・ワークロード、WL1 ワークロード、または WL3 ワークロードに割り当てられており、WC_DML ワーク・クラスと WC_LOAD ワーク・クラスに分類されるアクティビティーも、WA_MAP_DML および WA_MAP_LOAD ワーク・アクションが適用されます。つまり、作業タイプが LOAD のアクティビティーは、WA_MAP_LOAD ワーク・アクションにより SSC1 サービス・サブクラスにマップされ、作業タイプが DML のアクティビティーは、WA_MAP_DML ワーク・アクションにより SSC2 サービス・サブクラスにマップされます。

(定義されているどのワークロードとも接続属性が一致しないために) WL2 ワークロードまたはデフォルトのワークロードのいずれかに割り当てられたアクティビティーは、サービス・サブクラスに直接マップされます。特に、WL2 ワークロードはそのアクティビティーを SSC3 サービス・サブクラスにマップします。ワークロードがアクティビティーをサービス・サブクラスに直接マップする場合、それらのアクティビティーに適用されるワーク・アクションはありません。

ワーク・クラスの作業タイプおよび SQL ステートメント

ワーク・クラスには、関連する作業タイプがあります。 1 つ以上のアクティビティー・タイプを 1 つのワーク・クラス・タイプに分類することができます。ワーク・クラスを定義する前に、どのタイプのアクティビティーが各タイプのワーク・クラスに分類されるかを理解しておく必要があります。

以下の表は、ワーク・クラスに使用可能なタイプ・キーワードと、さまざまなキーワードに対応する SQL ステートメントを示しています。 load コマンドを除いて、以下の表内のすべてのステートメントは、EXECUTE、EXECUTE IMMEDIATE、または OPEN 要求の処理の直前に代行受信されます。ロード・ユーティリティーは、クライアントから実行した場合、データ・サーバー上で実際のロード操作を開始する前に要求を発行することがあります。

表 13. 作業タイプ・キーワードおよび関連 SQL ステートメント

作業タイプ・キーワード	該当する SQL ステートメント
READ	• すべての SELECT ステートメント (SELECT INTO、VALUES INTO、全選択) **注:** DELETE、INSERT、または UPDATE を含む SELECT ステートメントは除外されます。 • すべての XQuery ステートメント

表 13. 作業タイプ・キーワードおよび関連 SQL ステートメント (続き)

作業タイプ・キーワード	該当する SQL ステートメント
WRITE	• すべての UPDATE ステートメント (検索条件付き、位置指定) • すべての DELETE ステートメント (検索条件付き、位置指定) • すべての INSERT ステートメント (values、副選択) • すべての MERGE ステートメント • DELETE、INSERT、または UPDATE ステートメントを含むすべての SELECT ステートメント
CALL	CALL ステートメント。 **注:** CALL ステートメントは、CALL および ALL ワーク・クラス・タイプにのみ分類されます。
DML	READ および WRITE ワーク・クラス・タイプに分類される全ステートメント。
DDL	• すべての ALTER ステートメント • すべての CREATE ステートメント • COMMENT ステートメント • DECLARE GLOBAL TEMPORARY TABLE ステートメント • DROP ステートメント • FLUSH PACKAGE CACHE ステートメント • すべての GRANT ステートメント • REFRESH TABLE • すべての RENAME ステートメント • すべての REVOKE ステートメント • SET INTEGRITY ステートメント
LOAD	ロード・ユーティリティー **注:** ロード・ユーティリティーは、LOAD および ALL ワーク・クラス・タイプにのみ分類されます。

表 13. 作業タイプ・キーワードおよび関連 SQL ステートメント (続き)

作業タイプ・キーワード	該当する SQL ステートメント
ALL	すべてのデータベース・アクティビティー。 **注:** アクションがしきい値の場合、しきい値が適用されるデータベース・アクティビティーは、しきい値のタイプに応じて異なります。例えば、しきい値のタイプが ESTIMATEDSQLCOST の場合、(timeron 単位の) 見積コストがある DML アクティビティーだけがこのしきい値により影響を受けます。 詳しくは、 178 ページの『例: ALL キーワードを使用して定義されたワーク・クラスの処理』を参照してください。

以下の図は、作業タイプ・キーワードの階層図を示しています。

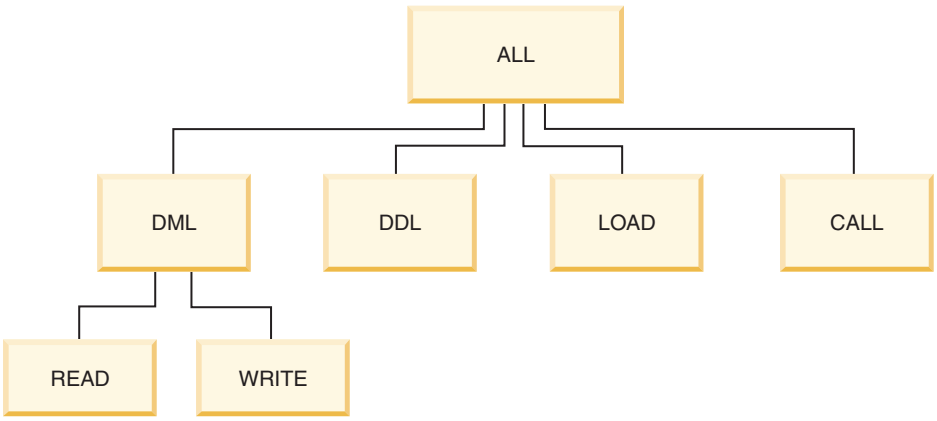


図 16. 作業タイプ・キーワード

使用可能などのキーワードにも属さない SQL ステートメントは分類されず、ワーク・クラスやワーク・クラス・セットが存在しないかのように振る舞います。例えば、ステートメントが SET SCHEMA であり、ワーク・クラス・セット内の唯一のワーク・クラスが作業タイプ DML である場合、そのステートメントは分類されず、適用できるワーク・アクションはありません。それで、アクションが MAP の場合、SET SCHEMA アクティビティーはデフォルトのサービス・サブクラス (SYSDEFAULTSUBCLASS) で実行します。アクションがしきい値の場合、アクティビティーに適用されるしきい値はありません。

ワーク・クラス・セット内のワーク・クラスの評価順序

ワーク・クラス・セットは、データベース・アクティビティーと一致する複数のワーク・クラスを持つことができます。ワーク・クラス・セットからアクティビティーの分類先となるワーク・クラスを選択するために、データ・サーバーは評価順序に応じてワーク・クラスを評価し、アクティビティーと一致する最初のワーク・クラスで停止します。

一致するワーク・クラスが存在しない場合、データベース・アクティビティーはどのワーク・クラスにも属さず、そのアクティビティーに適用されるワーク・アクションはありません。適用されるワーク・アクションがあるワーク・クラスだけが考慮されます。

ワーク・クラス・セットの作成時または変更時に、ワーク・クラス・セット内のワーク・クラスの評価順序を調整することができます。ワーク・クラス・セットの作成時または変更時には、以下の 3 つの方法の 1 つを使用して、ワーク・クラス・セット内でのワーク・クラスの位置を決めます。

- リスト内でワーク・クラスの絶対位置を指定します。

  例えば、POSITION AT 2 とすることができます。この場合、ワーク・クラスはワーク・クラス・セット内の 2 番目の位置に配置され、2 番目の位置にあったワーク・クラスは 3 番目の位置に、3 番目のワーク・クラスは 4 番目の位置に、という具合になります。 CREATE WORK CLASS SET または ALTER WORK CLASS SET ステートメントによりワーク・クラスに指定された位置が、ワーク・クラス・セット内のワーク・クラスの合計数より大きい数値の場合、ワーク・クラスはリストの最後に位置します。
- POSITION BEFORE または POSITION AFTER キーワードを使用して、既にワーク・クラス・セット内にあるワーク・クラスとの相対的なワーク・クラスの位置を指定します。
- ワーク・クラスの作成時に位置を省略します。

  この状況では、新規ワーク・クラスはリストの末尾に配置されます。ワーク・クラス・セットのリスト内のワーク・クラスに指定する位置は、必ずしも SYSCAT.WORKCLASSES ビュー内の EVALUATIONORDER 列の実際の値ではありません。データ・サーバーは、ギャップを避けるために順序値を自動的に割り当てます。

ワーク・クラスは受け取られる順序で処理されますが、これは評価順序に影響を与える可能性があります。例えば、以下のステートメントを発行すると仮定します。

```
ALTER WORK CLASS SET WCS ALTER WORK CLASS C1 POSITION AT 1
ALTER WORK CLASS C2 POSITION AT 1
```

この結果、C1 ワーク・クラスは評価順序 2 を持ち、C2 ワーク・クラスは評価順序 1 を持ちます。処理された最後のワーク・クラスは C2 だったためです。

ワーク・アクションとワーク・アクション・セットのドメイン

データベースまたはサービス・スーパークラスのいずれかにワーク・アクション・セットを定義できます。ワーク・アクション・セットに定義できるワーク・アクションのタイプは、ワーク・アクション・セットの定義対象のオブジェクトのタイプに応じて異なります。

ワーク・アクション・セットがデータベースに定義される場合、ワーク・アクション・セット内のワーク・アクションは以下のアクションの 1 つでなければなりません。

- しきい値

実際のしきい値は WHEN threshold-type キーワードにより指定されます。複数のしきい値のワーク・アクションは、すべてのしきい値が異なるタイプである場合は単一のワーク・クラスに適用できます。このアクションが指定される場合、しきい値はワーク・クラスに関連付けられたすべてのデータベース・アクティビティーに適用されます。

- PREVENT EXECUTION

  このアクションが指定された場合、関連付けられたワーク・クラスに一致するすべてのデータベース・アクティビティーは実行を許可されません。

- COLLECT ACTIVITY DATA

  このアクションが指定された場合、このワーク・アクションが定義されているワーク・クラスに対応するデータベース・アクティビティーに関する情報は、アクティビティーの実行完了時に、アクティブな ACTIVITIES イベント・モニターに書き込まれます。詳しくは、 150 ページの『個々のアクティビティーのデータ収集』を参照してください。

- COUNT ACTIVITY

  このアクションが指定された場合、関連するワーク・クラスにマップするすべてのデータベース・アクティビティーにより、そのワーク・クラス・タイプの回転カウンターが増分されます。 (ワーク・クラスの回転カウンターは、アクティビティーがそのワーク・クラスに関連付けられるごとに 1 ずつ増分されます。) COUNT ACTIVITY ワーク・アクションにより、このカウンターは効率的な仕方で更新されます。ワーク・クラスに対応するアクティビティーに適用されるワーク・アクションがない場合、ワーク・クラス・アクティビティー・カウンターは増分されません。時々注意すべきアクションは、特定のタイプのアクティビティーのカウントの入手だけということもあります。詳しくは、 150 ページの『個々のアクティビティーのデータ収集』を参照してください。

ワーク・アクション・セット内のワーク・アクションがこれらのアクションのいずれでもない場合、SQL4720N が返されます。

サービス・スーパークラスにワーク・アクション・セットを定義する場合、ワーク・アクション・セット内のワーク・アクションは以下のアクションの 1 つでなければなりません。

- マッピング・アクション

  アクティビティーは、デフォルトのサービス・サブクラスを除き、サービス・スーパークラス内のどのサービス・サブクラスにでもマップできます。 MAP ACTIVITY TO SERVICE CLASS キーワードを使用して、アクティビティーのマップ先となるサービス・サブクラスを指定します。ワーク・アクション・セット内の 1 つのマップ・ワーク・アクションだけを、同じワーク・クラスに適用することができます。

- PREVENT EXECUTION

  振る舞いはデータベース・ワーク・アクションと同じです。

- COLLECT ACTIVITY DATA

  振る舞いはデータベース・ワーク・アクションと同じです。

- COLLECT AGGREGATE ACTIVITY DATA

  このアクションが指定された場合、このワーク・アクションの定義対象のワーク・クラスに対応する集約データベース・アクティビティー・データが収集されます。
- COUNT ACTIVITY

  振る舞いはデータベース・ワーク・アクションと同じです。

ワーク・アクション・セット内のワーク・アクションがこれらのアクションのいずれでもない場合、SQL4720N が返されます。

以下の図は、LARGE ACTIVITIES と呼ばれるワーク・クラス・セット内のワーク・クラスが、データベースとサービス・スーパークラスの両方にどのように適用されるかの例を示しています。この目標に達するために、2 つのワーク・アクション・セット (Database large activities と Service class large activities) が作成されています。

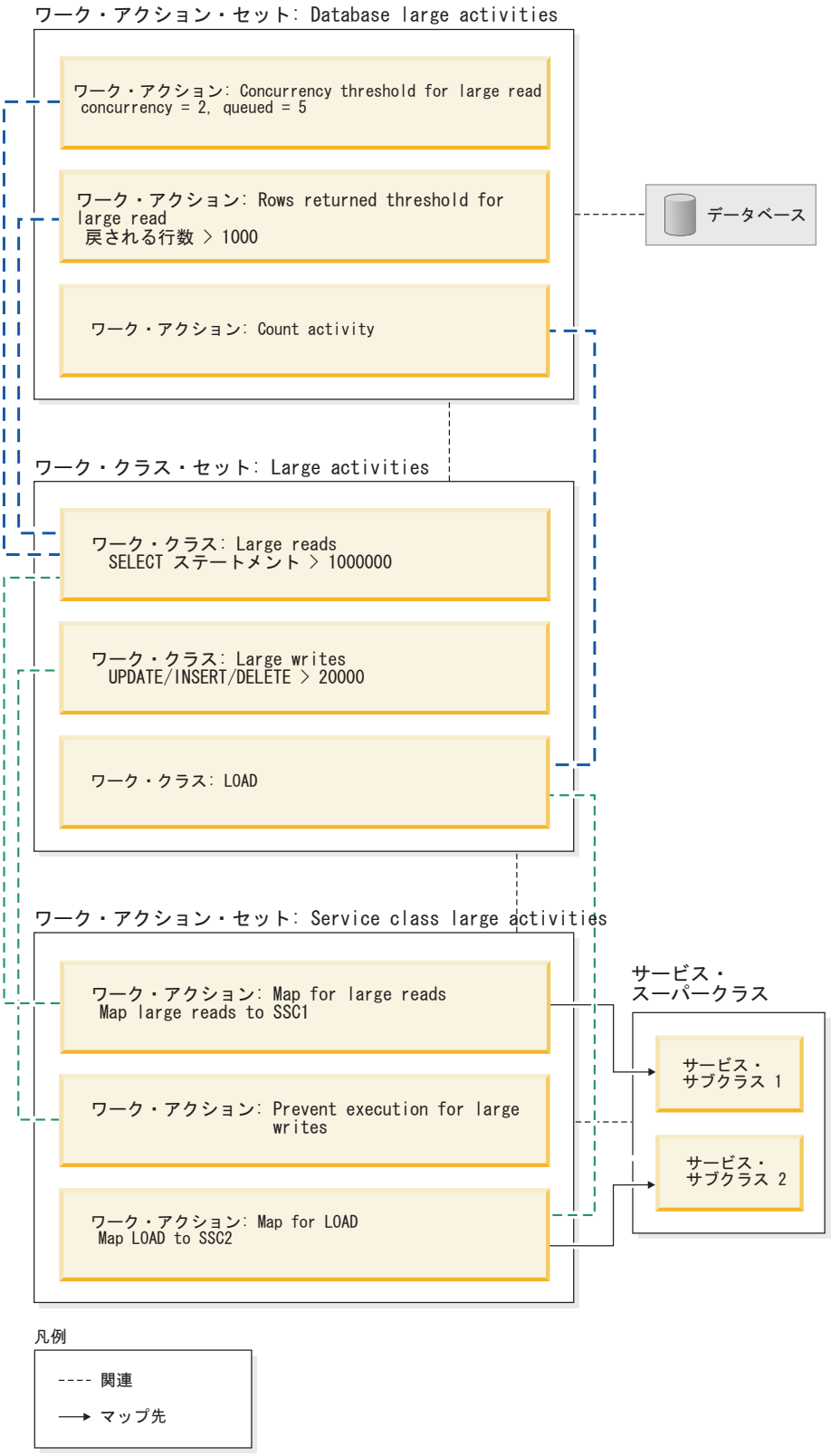


図 17. ワーク・アクション、ワーク・アクション・セット、ワーク・クラス、およびワーク・クラス・セットの例

ワーク・アクション・セットは以下のとおりです。

- Database large activities には以下が含まれています。

- Concurrency threshold for large reads。これにより 2 つの大規模な読み取りを同時に実行し、5 つの大規模読み取りをキューに入れることができます。
- Rows returned threshold for large reads。これにより大規模読み取りが 1000 を超える行を返すことはなくなります。
- Count activity for load。これはロード・ユーティリティーがデータベースに対して実行される回数をカウントします。

• Service class large activities には以下が含まれています。
- Map for large reads。これは大規模読み取りをサービス・サブクラス 1 にマップします。
- Map for large writes。これにより大規模書き込みが実行されることはなくなります。
- Map for LOAD。これはロードをサービス・サブクラス 2 にマップします。

ワーク・アクション・セットには、ワーク・アクション・セットが適用されるワーク・クラス・セット内のすべてのワーク・クラスに対してアクションが含まれている必要はありません。加えて、アクション・タイプが異なっていれば、複数のワーク・アクションをワーク・クラスに適用することができます。しきい値タイプが異なっていれば、複数のワーク・アクションをワーク・クラスに適用することができます。

ワーク・アクションで使用できるしきい値

データベースに定義されたワーク・アクション・セットには、しきい値を指定するワーク・アクションを含めることができます。

以下のしきい値がサポートされています。

• 集約しきい値:
- CONCURRENTDBCOORDACTIVITIES

• アクティビティーしきい値:
- SQLTEMPSPACE
- SQLROWSRETURNED
- ACTIVITYTOTALTIME
- ESTIMATEDSQLCOST

しきい値ごとにサポートされる作業の分類

ワーク・アクションで使用できるしきい値タイプはどれも任意のワーク・クラスに関連付けできますが、すべてのタイプのデータベース・アクティビティーがこれらしきい値タイプのすべてについてサポートされるわけではありません。

例えば、DDL のワーク・クラスを作成する場合、そのワーク・クラスを ESTIMATEDSQLCOST しきい値ワーク・アクションと関連付けます。このしきい値は、DDL に分類される要求のいずれにも適用されません。これは、DDL ステートメントが見積もりコストを持たないためです。 ALL のワーク・クラスを作成する場合、そのワーク・クラスを ESTIMATEDSQLCOST しきい値ワーク・アクション

に関連付けます。すべてのデータベース・アクティビティーは ALL ワーク・クラスに属しますが、しきい値は見積もりコストを持つデータベース・アクティビティーにのみ適用されます。

次の表に、どのワーク・クラス・カテゴリーがどのしきい値タイプでサポートされるかを示します。

表 14. しきい値ごとにサポートされる作業の分類

	75 ページの『CONCURRENTDBCOORDACTIVITIES しきい値』	67 ページの『SQLTEMPSPACE しきい値』	68 ページの『SQLROWSRETURNED しきい値』	67 ページの『ESTIMATEDSQLCOST しきい値』	69 ページの『ACTIVITYTOTALTIME しきい値』
READ	可	可	可	可	可
WRITE	可	可	可	可	可
CALL	可	不可	不可 (注を参照)	不可	可
DML	可	可	可	可	可
DDL	可	不可	不可	不可	可
LOAD	可	不可	不可	不可	可
ALL	可	一部可	一部可	一部可	可

注: 呼び出されたプロシージャーのステートメントから行が戻されることがありますが、これらの行は CALL ステートメントの結果として戻されるわけではないため、SQLROWSRETURNED しきい値による制御は受けません。

ワーク・クラスへのアクティビティーの割り当て

ワーク・クラス・セットが、ワーク・アクション・セットを経てデータベースまたはサービス・スーパークラスのいずれかに関連付けられている場合、実行、即時に実行、またはオープン要求の処理の実行前、あるいはロード・ユーティリティーの実行直前に、データベース・アクティビティーはワーク・クラス・セット内のワーク・クラスで指定されたいずれかの基準に一致するかどうかを判別するために検査されます。

ワーク・クラスは、ワーク・クラス・セット内でその評価順序によってソートされます。この評価順序に基づいて、データベース・アクティビティーはデータベース・アクティビティーの属性 (アクティビティー・タイプやカーディナリティーなど) に基づく各ワーク・クラスに照らして検査されます。これは一致するものがあるか、またはワーク・クラス・セット内のワーク・クラスのリストの末尾に達するまで続けられます。

以下のワーク・クラスがワーク・クラス・セット内にあり、すべてのワーク・クラスがそれに適用されるワーク・アクションを持っていると仮定します。

- 評価順序: 1。ワーク・クラス名: `MyLoad`。ワーク・クラス・タイプ: LOAD
- 評価順序: 2。ワーク・クラス名: `SmallRead`。ワーク・クラス・タイプ: READ。他の属性: 見積コスト < 300 timeron
- 評価順序: 3。ワーク・クラス名: `AllDML`。ワーク・クラス・タイプ: DML
- 評価順序: 4。ワーク・クラス名: `LargeRead`。ワーク・クラス・タイプ: READ。他の属性: 見積コスト > 301 timeron
- 評価順序: 5。ワーク・クラス名: `MyDDL`。ワーク・クラス・タイプ: DDL

見積コスト 200 timeron の SELECT ステートメントを受け取る場合、それは `SmallRead` ワーク・クラスに割り当てられます。 DDL アクティビティー (CREATE TABLE など) が着信する場合、それには `MyDDL` ワーク・クラスが割り当てられま

す。見積コスト 500 timeron の SELECT ステートメントを受け取る場合、AllDML は LargeRead ワーク・クラスの前に位置するので、それは AllDML ワーク・クラスに割り当てられます。詳しくは、 178 ページの『例: ALL キーワードを使用して定義されたワーク・クラスの処理』を参照してください。

ワーク・クラスにそれに適用されるワーク・アクションがない場合、そのワーク・クラスは無視され、割り当てられるアクティビティーはありません。詳しくは、 96 ページの『ワーク・クラス・セット内のワーク・クラスの評価順序』を参照してください。

データベース・アクティビティーに対するワーク・アクションの適用

1 つのデータベースまたはサービス・スーパークラスに対して 1 つのワーク・アクション・セットのみを適用できます。

データ・サーバーにサブミットされると、作業はワークロード (ユーザー定義のワークロードまたはデフォルトのワークロードのいずれか) に関連付けられ、続いてサービス・クラスにマップされます。

以下の図は、ワーク・アクションをアクティビティーに適用する方法のプロセスについて示しています。

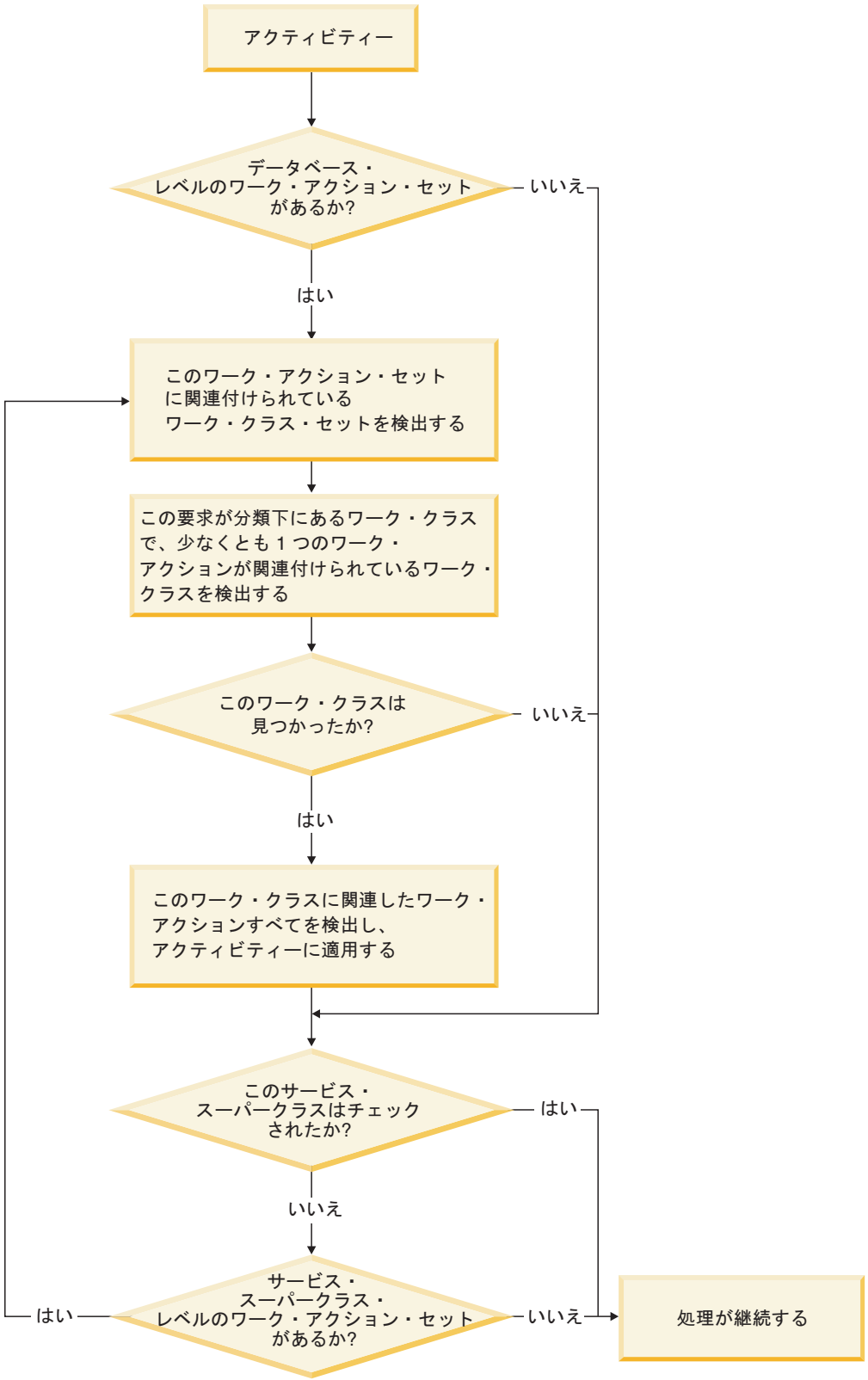


図 18. アクティビティーに対するワーク・アクションの適用

ワーク・アクションは、以下のようにアクティビティーに割り当てられます。

1. アクティビティーがサービス・スーパークラスまたはサービス・サブクラスにマップされると、データ・サーバーは使用可能なデータベース・レベルのワーク・アクション・セットが存在するかどうかを調べます。
2. 使用可能なデータベース・レベルのワーク・アクション・セットが存在する場合、データ・サーバーは、データベース・レベルのワーク・アクション・セットが関連付けられているワーク・クラス・セットのいずれかのワーク・クラスにアクティビティーが該当しているかどうかを調べます。
3. アクティビティーが、1 つ以上のワーク・アクションが適用されているワーク・クラスに該当した場合、それらのワーク・アクションがアクティビティーに適用されます。
4. 次に、アクティビティーがワークロードによってサービス・スーパークラスにマップされる場合、データ・サーバーはワーク・アクション・セットがサービス・スーパークラスに適用されるかどうかを調べます。
5. ワーク・アクション・セットがサービス・スーパークラスに適用される場合、データ・サーバーは、サービス・スーパークラス・レベルのワーク・アクション・セットが関連付けられているワーク・クラス・セットのいずれかのワーク・クラスにアクティビティーが該当しているかどうかを調べます。
6. アクティビティーが、1 つ以上のワーク・アクションが適用されているワーク・クラスに該当した場合、それらのワーク・アクションがアクティビティーに適用されます。

以下の場合、アクティビティーはワーク・アクション・セットによる影響を受けません。

- アクティビティーがデフォルト・システム (SYSDEFAULTSYSTEMCLASS) およびデフォルト保守 (SYSDEFAULTMAINTENANCECLASS) サービス・クラスに該当する場合。
- アクティビティーがデフォルトの管理ワークロード SYSDEFAULTADMWORKLOAD に割り当てられている場合。
- アクティビティーがロード操作の内部にある場合。ロード操作自体は、ワーク・アクション・セットの評価を受けます。
- システムのストアード・プロシージャーの子アクティビティーの場合。唯一の例外は、SYSPROC.ADMIN_CMD ストアード・プロシージャーです。SYSPROC.ADMIN_CMD の子アクティビティーは、ワーク・アクション・セットの評価を受けます。
- ワーク・アクション・セットが使用不可になっている場合。
- ワークロードがアクティビティーをサービス・サブクラスに直接マップする場合。

ワークロードとワーク・アクション・セットの比較

データベース・アクティビティーに対してどのようなタイプの制御を保持したいかに応じて、ワークロードのみを使用して、あるいはワークロードとワーク・アクションの両方を使用して、アクティビティーをサービス・クラスにマップすることができます。

ワークロードを使用した場合、要求の識別とサービス・クラスへの割り当ては接続属性に基づいて行われます。ワークロードは、作業を特定の DB2 サービス・クラスにルーティングして実行するための主な方法です。要求を識別する方法をさらに詳細化する必要がある場合は、ワーク・クラスを使用して、そのタイプおよび他のアクティビティー属性に基づいてアクティビティーを分類することができます。例えば、READ アクティビティー、WRITE アクティビティー、および LOAD アクティビティーを異なるワーク・クラスに分類することができます。

ワーク・クラス (ワーク・クラス・セットにグループ化された) を使用する場合、ワーク・アクションを使用してさまざまなタイプのアクティビティーを制御することができます。例えば、1 つのワーク・アクションを使用して、ある特別なタイプのアクティビティーを、あるサービス・クラスにマップすることができ、別のワーク・アクションを使用して、同じタイプのアクティビティーが特定の条件を超えないよう、しきい値を適用することができます。

ワーク・アクションはワーク・アクション・セットにグループ化されます。単一のワーク・アクション・セットは、データベース内のアクティビティーまたはサービス・スーパークラス内のアクティビティーに適用できます (両方は不可)。ワーク・クラス・セットとワーク・アクション・セットは一緒に動作します。つまり、アクティビティーを特定のタイプの作業として分類するためのワーク・クラスがあらかじめ存在していなければ、ワーク・アクションをワーク・クラスに適用することはできません。 1 つのワーク・クラス・セットは複数のワーク・アクション・セットに関連付けることができますが、1 つのワーク・アクション・セットは 1 つのワーク・クラス・セットにだけ関連付けることができます。

以下の図は、ワークロードおよびワーク・アクション・セットを使用するワークロード管理のインプリメンテーションの例を示しています。この図では、要求をサブミットしたユーザー ID に基づいて、要求がワークロード WL_A に割り当てられると仮定します。ワークロード WL_A は、要求がサービス・スーパークラス SC_A で実行されることを指定します。ワーク・クラス・セット WCS_1 のワーク・クラスは、ワークロード WL_A に関連した要求が実行する作業タイプと一致すると仮定します。

例えば、カタログを更新しないアクティビティー (READ アクティビティー) がシステムに入ってくると仮定します。データベース・レベルのワーク・アクション・セット WAS_1 (ワーク・クラス・セット WCS_1 と関連付けられている) には、READ ワーク・クラスに適用されるワーク・アクションが含まれており、500 以内のアクティビティーであればデータベース全体で同時に実行可能とする、しきい値を設定します。要求がこのしきい値で設定された境界を超えないと仮定すると、要求はサービス・スーパークラス SC_A に (ワークロード WL_A によって) マップされます。ここで要求は、サービス・スーパークラス・レベルのワーク・アクション・セット WAS_2 を検出します。これもまたワーク・クラス・セット WCS_1 に関連付けられており、サービス・スーパークラス SC_A 内のアクティビティーに適用されます。このワーク・アクション・セットには、マッピング・ワーク・アクションが含まれており、これも READ ワーク・クラスに適用され、これによってすべての READ アクティビティーは、サービス・スーパークラス SC_A 内のサービス・サブクラス SSC_1a にマップされます。

これにやや似た状態が、ワークロード WL_B に関連付けられた要求 (この場合もその接続属性に基づいて関連付けられる) でも発生します。ワークロード WL_B はアクティビティーをサービス・スーパークラス SC_B にマップします。この要求は LOAD アクティビティーに関するものであり、ワーク・クラス・セット WCS_2 には LOAD アクティビティーに当てはまるワーク・クラスが含まれていると仮定します。ワーク・クラス・セット WCS_2 はサービス・スーパークラス・レベルのワーク・アクション・セット WAS_3 に関連付けられ、これはサービス・スーパークラス SC_B のアクティビティーに適用されます。ワーク・アクション・セット WAS_3 に、LOAD ワーク・クラスに適用されているマッピング・ワーク・アクションが含まれていると仮定すると、LOAD アクティビティーがワークロード WL_B によってサービス・スーパークラス SC_B にマップされる場合、それはワーク・アクションによって実行のためにサービス・サブクラス SSC_1b にマップされます。

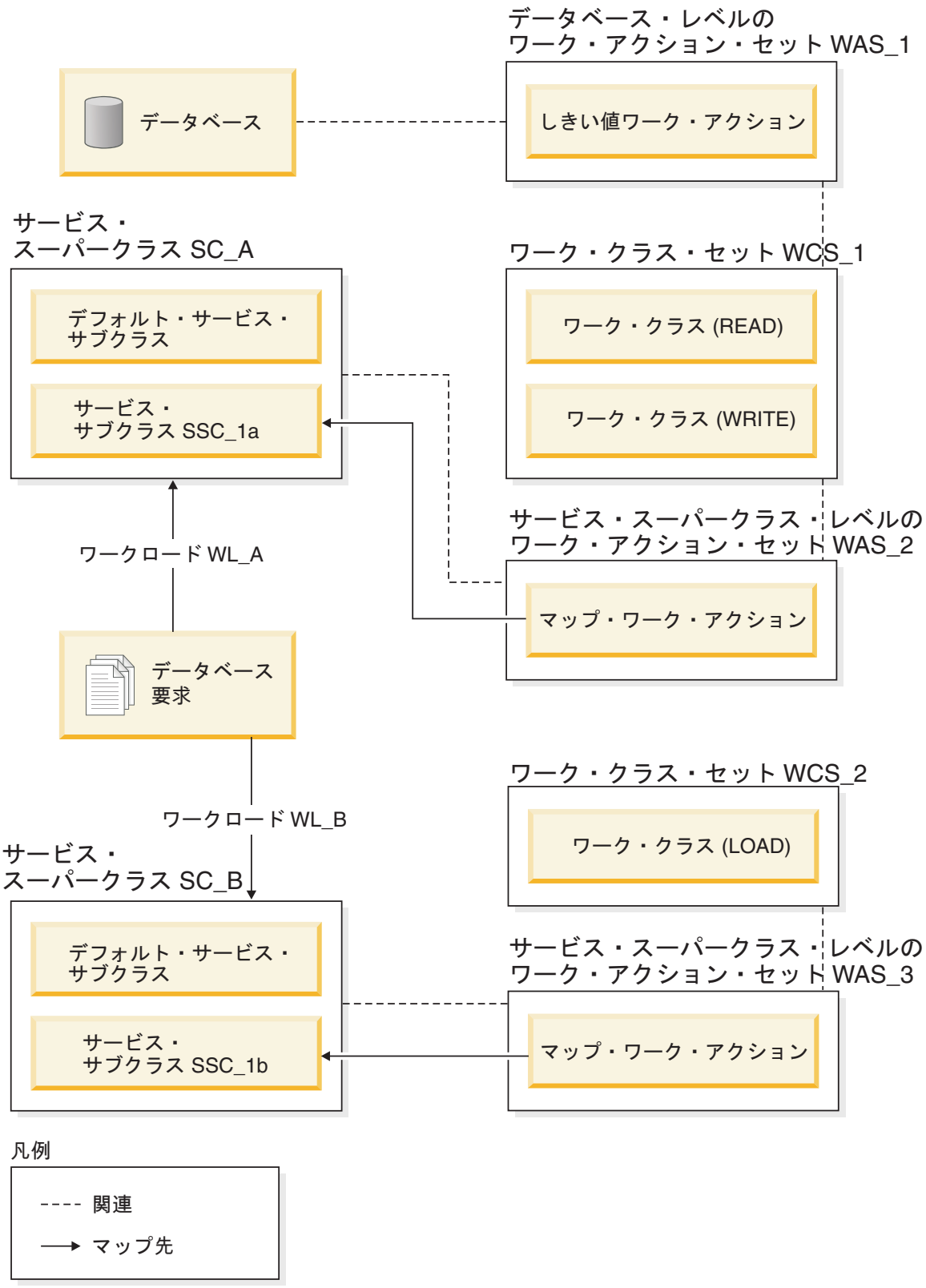


図 19. ワークロードとワーク・アクション・セット

ワーク・アクション・セットおよびワーク・アクションの作業

ワーク・アクション・セットの作成

ワーク・アクションおよびワーク・アクション・セットを作成するには、CREATE WORK ACTION SET ステートメントを使用します。

ワーク・アクション・セットを作成するためには、SYSADM または DBADM 権限が必要です。

その他の前提条件については、以下のトピックを参照してください。

- 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』
- 命名規則

ワーク・アクション・セットを作成するときは、次のようにします。

- ワーク・アクション・セットをワーク・クラス・セットに関連付けます。ワーク・クラス・セットが既に存在している必要があります。
- さらに、ワーク・アクション・セットをデータベースまたはサービス・スーパークラスとも関連付けます。ワーク・アクション・セットをサービス・スーパークラスに関連付けている場合、このサービス・クラスが既に存在している必要があります。ワーク・アクション・セットをシステム・サービス・クラス (SYSDEFAULTSYSTEMCLASS) または保守サービス・クラス (SYSDEFAULTMAINTENANCECLASS) に定義することはできません。

ワーク・アクション・セットを作成するには、次のようにします。

1. CREATE WORK ACTION SET ステートメントを次のオプションとともに使用します。
   - ワーク・アクション・セットの名前を指定。ワーク・アクション・セットの名前は、データベースの中で固有でなければなりません。
   - ワーク・アクション・セットが関連付けられるオブジェクトを指定。データベースまたはサービス・スーパークラスを指定できます。ワーク・アクション・セットをデータベースに関連付けるように指定した場合、そのワーク・アクション・セットに含まれるワーク・アクションは、いずれもマッピング・ワーク・アクションまたは収集集約アクションにはなれません。ワーク・アクション・セットをサービス・スーパークラスに関連付けるように指定した場合、そのワーク・アクション・セットに含まれるワーク・アクションは、いずれもしきい値にはなれません。例えば、ワーク・アクション・セットを REPORTS サービス・スーパークラスに適用するには、次のように指定します。

     ```
     FOR SERVICE CLASS REPORTS
     ```

     ワーク・アクション・セットをデータベースに適用するには、次のように指定します。

     ```
     FOR DATABASE
     ```

   - ワーク・アクション・セットが関連付けられるワーク・クラス・セットを指定。ワーク・クラス・セットのワーク・クラスによって、ワーク・アクション・セット内のワーク・アクションが適用されるデータベース・アクティビティーが分類されます。例えば、ワーク・アクション・セットを LARGEREADS ワーク・クラス・セットに関連付けるには、次のように指定します。

```
USING WORK CLASS SET LARGEREADS
```

- オプション: ワーク・アクション・セットに 1 つ以上のワーク・アクションを作成。手順については、 112 ページの『ワーク・アクションの作成』を参照してください。
- ワーク・アクション・セットを使用可能にするか使用不可にするかを指定。デフォルトでは、ワーク・アクション・セットは使用可能です。ワーク・アクション・セットが使用不可の場合、データ・サーバーはアクティビティーの実行時にこのワーク・アクション・セット (またはそれに含まれるすべてのワーク・アクション) を考慮しません。

2. 変更をコミットします。 変更をコミットすると、ワーク・アクション・セットが SYSCAT.WORKACTIONSETS ビューに追加されます。

   新しいワーク・アクション・セットがデータベースで有効になるのはコミットされた後で、現在実行中のどのデータベース・アクティビティーにも影響しません。

ワーク・アクション・セットの変更

ワーク・アクション・セットに含まれるワーク・アクションを追加、変更、ドロップしたり、ワーク・アクション・セットを使用可能または使用不可にするには、ALTER WORK ACTION SET ステートメントを使用します。

ワーク・アクション・セットを変更するためには、SYSADM または DBADM 権限が必要です。

その他の前提条件については、以下のトピックを参照してください。

- 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』
- 命名規則

特定のワーク・クラス・セットと併用されるワーク・アクション・セットを作成するときは、別のワーク・クラス・セットと併用されるように変更することはできません。これは、ワーク・アクション・セットに含まれるワーク・アクションには、ワーク・クラス・セットに含まれるワーク・クラスと従属関係があるためです。このワーク・アクション・セットを適用するワーク・クラス・セットを変更する場合は、ワーク・アクション・セットをドロップして再作成する必要があります。

ワーク・アクション・セットに含まれるワーク・アクションのタイプはワーク・アクション・セットが定義されているオブジェクト (データベースまたはサービス・スーパークラス) に依存するため、ワーク・アクション・セットの適用先となるオブジェクトを変更することはできません。ワーク・アクション・セットが関連付けられる先のオブジェクトを変更する場合は、ワーク・アクション・セットをドロップして、再作成する必要があります。

ワーク・アクション・セットを変更するには、次のようにします。

1. ワーク・アクション・セットに新しいワーク・アクションを追加する場合は、ADD キーワードを使用します。 ワーク・アクション・セットにワーク・アクションを追加するときに指定できる各パラメーターについては、 112 ページの『ワーク・アクションの作成』を参照してください。

2. 既存のワーク・アクションを変更する場合は、ALTER キーワードを使用します。 ワーク・アクションを変更する方法については、 116 ページの『ワーク・アクションの変更』を参照してください。
3. ワーク・アクションをドロップする場合は、DROP キーワードを使用します。 ワーク・アクション・セットからワーク・アクションをドロップする方法については、 118 ページの『ワーク・アクションのドロップ』を参照してください。
4. 現在使用可能でないワーク・アクション・セットを使用可能にすることもできますし、その逆もできます。 使用可能なワーク・アクション・セットを使用不可にする場合、変更をコミットした後は、データ・サーバーはそのワーク・アクション・セットを無視します。詳しくは、『ワーク・アクション・セットを使用不可にする』を参照してください。ワーク・アクション・セットを使用可能にする場合、変更をコミットした後は、そのワーク・アクション・セットは、データベースに入力される、適用可能な次のアクティビティーに適用されます。
5. 変更をコミットします。 変更をコミットすると、ワーク・アクション・セットが SYSCAT.WORKACTIONSETS ビューで更新されます。 SYSCAT.WORKACTIONS ビューは、ワーク・アクションが追加、変更、またはドロップされると更新されます。

ワーク・アクション・セットを使用不可にする

ワーク・アクション・セットを使用不可にするには、CREATE WORK ACTION SET ステートメントまたは ALTER WORK ACTION SET ステートメントの DISABLE キーワードを使用します。

ワーク・アクション・セットを使用不可にするためには、SYSADM または DBADM 権限が必要です。

実行時に、使用不可にされたワーク・アクション・セットはあたかも存在しないかのようにして処理されます。例えば、READCLASSES というワーク・クラスに関連付けられた READACTIVITIES というワーク・アクション・セットがあり、そのワーク・アクション・セットが READSERVICECLASS というサービス・スーパークラスに定義されていると仮定します。 SMALLREAD ワーク・アクション・セットの中には、すべての SELECT ステートメントをサービス・サブクラス SMALLREADSERVICECLASS に再マップするワーク・アクションがあります。 READACTIVITIES ワーク・アクション・セットを使用不可にすると、すべての SELECT ステートメントはあたかも READACTIVITIES ワーク・アクション・セットが存在していないかのように処理されて、デフォルトのサービス・サブクラスにマップされます。

ワーク・アクション・セットを使用不可にするには、次のようにします。

1. ワーク・アクション・セットを作成するのか変更するのかに応じて、以下のステートメントのいずれかを使用します。
   - CREATE WORK ACTION SET ステートメントを使用して、ワーク・アクション・セットを使用不可にする。以下に例を示します。

     ```
     CREATE WORK ACTION SET work-action-set-name ... DISABLE
     ```

   - ALTER WORK ACTION SET ステートメントを使用する。以下に例を示します。

```
ALTER WORK ACTION SET work-action-set-name ... DISABLE
```

2. 変更をコミットします。 変更をコミットすると、ワーク・アクション・セットが SYSCAT.WORKACTIONSETS ビューで更新されます。

ワーク・アクション・セットのドロップ

ワーク・アクション・セットをドロップするには、DROP WORK ACTION SET ステートメントを使用します。

ワーク・アクション・セットをドロップするためには、SYSADM または DBADM 権限が必要です。

ワーク・アクション・セットをドロップすると、ワーク・アクション・セットおよびそれに含まれるすべてのワーク・アクションがドロップされます。

ワーク・アクション・セットに CONCURRENTDBCOORDACTIVITIES しきい値ワーク・アクションが含まれる場合、そのワーク・アクション・セットをドロップできるようにするには、まずそのワーク・アクションを使用不可にする必要があります。

ワーク・アクション・セットをドロップするには、次のようにします。

1. DROP WORK ACTION SET ステートメントを使用します。
2. 変更をコミットします。 変更をコミットすると、ワーク・アクション・セットが SYSCAT.WORKACTIONSETS ビューから除去されます。さらに、そのワーク・アクション・セットの一部だったすべてのワーク・アクションが、SYSCAT.WORKACTIONS ビューから除去されます。ワーク・アクション・セットにしきい値ワーク・アクションが含まれる場合、そのしきい値は SYSCAT.THRESHOLDS ビューから除去されます。

ワーク・アクションの作成

CREATE WORK ACTION SET ステートメントまたは ALTER WORK ACTION SET ステートメントを使用して、ワーク・アクションを作成します。

ワーク・アクションを作成するためには、SYSADM または DBADM 権限が必要です。

その他の前提条件については、以下のトピックを参照してください。

- 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』
- 命名規則

ワーク・アクションを作成するときは、次のようにします。

- ワーク・アクションをワーク・クラスに関連付けます。ワーク・クラスは、ワーク・アクション・セットが適用されるワーク・クラス・セット内に既に存在していなければなりません。
- ワーク・アクションがしきい値である場合、そのワーク・アクション・セットがデータベースに対して定義されていなければなりません。ワーク・アクションでサポートされるしきい値のリストは、101 ページの『ワーク・アクションで使用できるしきい値』を参照してください。

- マッピング・ワーク・アクションを作成する場合は、そのワーク・アクション・セットがサービス・スーパークラスに対して定義されていなければなりません。マップ先のサービス・サブクラスは、このワーク・アクション・セットが定義されるサービス・スーパークラス内に既に存在している必要があります。さらに、デフォルトのサービス・サブクラスを指定することはできません。
- 同じタイプのワーク・アクション 1 つのみを、同じワーク・アクション・セットからの同じワーク・クラスに適用できます。しきい値は例外です。複数のしきい値を 1 つのワーク・クラスに適用できます。ただし、それぞれのしきい値は異なるタイプでなければなりません。
- 集約アクティビティー・データ収集ワーク・アクションを作成する場合は、そのワーク・アクション・セットがサービス・スーパークラスに対して定義されていなければなりません。

ワーク・アクションを作成するには、次のようにします。

1. CREATE WORK ACTION SET ステートメントの *work-action-definition* キーワード、または ALTER WORK ACTION SET ステートメントの *work-action-definition* キーワードを使用します。 ワーク・アクションに、以下を 1 つ以上指定します。
   - ワーク・アクションの名前。ワーク・アクションの名前は、ワーク・アクション・セット内で固有でなければなりません。
   - このワーク・アクションが適用されるワーク・クラスの名前。このワーク・クラスは、このワーク・アクション・セットが関連付けられているワーク・クラス・セット内にあるワーク・クラスの 1 つである必要があります。例えば、このワーク・アクションをワーク・クラス LARGEDML に適用するには、以下のように指定します。

     ```
     ON WORK CLASS LARGEDML
     ```

   - このワーク・アクションのワーク・クラスと一致するアクティビティーに適用されるアクション。
     – このワーク・アクション・セットがサービス・スーパークラスに関連付けられている場合は、MAP ACTIVITY キーワードを指定すると、ワーク・アクションはアクティビティーをそのサービス・スーパークラス内のサービス・サブクラスにマップします。デフォルトでは、マッピング・ワーク・アクションにより、ネストされたアクティビティーは、親と同じサービス・サブクラスにマップされます。ルーチン内で開かれたカーソルは、ネストされたアクティビティーの一例です。

       例えば、ワーク・アクションがサービス・サブクラス SMALLREAD にマップされるようにし、すべてのネストされたアクティビティーも同じサービス・サブクラスにマップされるようにする場合は、次のように指定します。

       ```
       MAP ACTIVITY TO SMALLREAD
       ```

       次のように指定することもできます。

       ```
       MAP ACTIVITY WITH NESTED TO SMALLREAD
       ```

ワーク・アクションがこのサービス・サブクラスにマップされるようにし、ネストされたアクティビティーはそのサービス・サブクラスにマップされないようにする場合は、次のように指定します。

```
MAP ACTIVITY WITHOUT NESTED TO SMALLREAD
```

ワーク・アクションを WITHOUT NESTED として定義した場合、ネストされたアクティビティーは親アクティビティーと同じサービス・サブクラスに自動的にマップされるのではなく、それら自体のアクティビティー・タイプに従って処理されるようになります。例えば、CALL アクティビティーがサービス・サブクラス `subsc1` にマップされ、このルーチン内にオープン・カーソルがあるとします。このオープン・カーソルが、別のマッピング・ワーク・アクションが適用される別のワーク・クラスに分類される場合、このオープン・カーソルは別のサービス・サブクラスへマップされます。

- このワーク・アクション・セットがデータベースに関連付けられている場合は、WHEN キーワードを指定して、アクティビティーにしきい値を適用し、アクティビティーによりしきい値の違反が生じた場合に処置が取られるようにすることができます。ワーク・アクションには、以下のしきい値を指定できます。
  - CONCURRENTDBCOORDACTIVITIES およびその QUEUEDACTIVITIES キーワード。
  - SQLTEMPSPACE
  - SQLROWSRETURNED
  - ESTIMATEDSQLCOST
  - ACTIVITYTOTALTIME

    注: ACTIVITYTOTALTIME しきい値に指定できる値の最大値は 2147483400 秒です。2147483400 秒よりも大きな値が (DAY、HOUR、MINUTE、または SECOND キーワードを使用して) 指定された場合は、すべて最大値で切り捨てられます。

  しきい値の違反が生じた場合に以下の処置が取られることを指定できます。
  - しきい値の違反の原因となったアクティビティーに関するアクティビティー・データを収集するかどうか。収集する場合、アクティビティー・データはアクティビティーの実行完了時に、アクティブなアクティビティー・イベント・モニターに書き込まれます。デフォルトでは、アクティビティーに関するデータは収集されません。このアクティビティーに関するデータを収集する場合には、コーディネーター・パーティション、特定のデータベース・パーティション、またはすべてのデータベース・パーティションから収集することができます。このデータの収集には、ステートメントおよびそのコンパイル環境に関する詳細を含めるオプションと含めないオプションがあります。ステートメントとコンパイル環境についての詳細を収集する場合は、アクティビティーに使用された入力データ値も指定できます。
  - しきい値の違反の原因となったアクティビティーの実行継続を許可するかどうか。デフォルトでは、アクティビティーは停止されます。

例えば、ワーク・アクションに、2,000 timeron を超えるコストの DML ステートメントをチェックさせ、しきい値の違反が起こった場合にそのアクティビティーに関する基本的なデータを収集して、実行を継続させる場合には、以下のように指定します。

```
WHEN ESTIMATEDSQLCOST > 2000 COLLECT ACTIVITY DATA CONTINUE
```

- このワーク・アクションに定義されたワーク・クラスに対応するアクティビティーが実行されないようにするには、PREVENT EXECUTION キーワードを使用できます。
- 別のアクション (データの収集やアクティビティーのマッピングなど) でオーバーヘッドをさらに課すことなく、このワーク・クラスに関連したデータベース・アクティビティー数をカウントするには、COUNT ACTIVITY キーワードを指定できます。
- このワーク・クラス下に分類されるアクティビティーのアクティビティー・データを収集するには、COLLECT ACTIVITY DATA キーワードを指定します。収集する場合、アクティビティー・データはアクティビティーの実行完了時に、アクティブなアクティビティー・イベント・モニターに書き込まれます。デフォルトでは、アクティビティーに関するデータは収集されません。このアクティビティーに関するデータを収集する場合には、コーディネーター・パーティションまたはすべてのデータベース・パーティションから収集することができます。ステートメントおよびコンパイル環境情報といったアクティビティーの詳細を収集する場合は、WITH DETAILS キーワードを指定してそうすることができます。AND VALUES キーワードを使用して、入力データ値 (入力データ値を持つアクティビティーについて) をアクティビティー・イベント・モニターに送信することもできます。

  例えば、サービス・スーパークラスに適用されるワーク・アクション・セットがあるとします。このワーク・アクションに割り当てられたすべてのアクティビティーについて、アクティビティー・データを該当するイベント・モニターに書き込むようにします。このデータには、すべての集約アクティビティー情報、コンパイル環境に関する情報、および入力データ値があればそれも含めます。以下のように指定します。

  ```
  COLLECT ACTIVITY DATA ON ALL WITH DETAILS AND VALUES
  ```

- このワーク・クラスに分類されるアクティビティーの集約アクティビティー・データを収集するには、COLLECT AGGREGATE ACTIVITY DATA キーワードを指定します。収集する場合、集約アクティビティー・データはキャプチャーされて該当するイベント・モニターへ送信されます。この情報は、**wlm_collect_int** データベース構成パラメーターで指定されたインターバルにより周期的に収集されます。

  例えば、サービス・スーパークラスに適用されるワーク・アクション・セットがあるとします。このワーク・アクションに割り当てられたすべてのアクティビティーについて、該当するイベント・モニターに集約アクティビティー・データを書き込むようにします。このデータには、基本データ、アクティビティー・データ操作言語 (DML) の見積コストのヒストグラム、およびアクティビティー DML の到着間隔時間のヒストグラムが含まれます。以下のように指定します。

  ```
  COLLECT AGGREGATE ACTIVITY DATA EXTENDED
  ```

- 対応するワーク・クラスに対して作成されるヒストグラムを記述するために、COLLECT AGGREGATE ACTIVITY DATA ワーク・アクションによって使用されるヒストグラム・テンプレート。ワーク・アクションによって使用されるヒストグラム・テンプレートを指定すると、SYSCAT.HISTOGRAMTEMPLATEUSE (そのサービス・クラスまたはワーク・アクションによって参照されるヒストグラム・テンプレートを表示するビュー) 内に対応する行が追加されます。例えば、デフォルトの到着間隔ヒストグラム・テンプレートの到着間隔統計を収集する場合、以下のように指定します。

  ```
  INTERARRIVALTIME HISTOGRAM TEMPLATE SYSDEFAULTHISTOGRAM
  ```

  ヒストグラムおよびヒストグラム・テンプレートについて詳しくは、140 ページの『ワークロード管理のヒストグラム』を参照してください。
- ワーク・アクションを使用可能にするか使用不可にするか。デフォルトで、ワーク・アクションは使用可能な状態で作成されますが、ENABLE キーワードまたは DISABLE キーワードを使用して、使用可能にするか使用不可にするかを指定できます。ワーク・アクションが使用不可にされると、アクティビティーがデータベースまたはサービス・スーパークラス (どのオブジェクトに対してワーク・アクション・セットを作成したかによる) に入るときに、データ・サーバーはこのワーク・アクションを無視します。

2. 変更をコミットします。 変更をコミットすると、ワーク・アクションは SYSCAT.WORKACTIONS ビューに追加されます。ワーク・アクションがしきい値である場合、そのしきい値は SYSCAT.THRESHOLDS ビューに追加されます。

   新規ワーク・アクションは、コミットされた後にはじめてデータベースで有効になり、現在実行中のデータベース・アクティビティーには影響しません。

ワーク・アクションの変更

ワーク・アクションを変更する必要がある場合には、ALTER WORK ACTION SET ステートメントを使用します。

ワーク・アクションを変更するためには、SYSADM または DBADM 権限が必要です。

その他の前提条件は、313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』を参照してください。

ワーク・アクションを変更するには、次のようにします。

1. ALTER WORK ACTION SET ステートメントの ALTER キーワードを使用して、ワーク・アクションの以下の特性 (複数可) を変更します。
   - ワーク・アクションが適用されるワーク・クラスを変更することができます。ワーク・クラスは、ワーク・アクション・セットが適用されるワーク・クラス・セット内に既に存在していなければなりません。
   - ワーク・アクションがサービス・サブクラスにマップする場合、どのサービス・サブクラスにデータベース・アクティビティーがマップされるかを変更することができます。マッピングは、同じサービス・スーパークラス内のサービス・サブクラスにのみ変更できます。デフォルトのサービス・サブクラスへは

マップできません。アクティビティー内のネストされたアクティビティーが同じサービス・サブクラスへマップされるかどうかを変更することもできます。例えば、ワーク・アクションが現在 WITH NESTED として定義されている場合、これを WITHOUT NESTED に変更することができます。この変更により、ネストされたアクティビティーは親アクティビティーと同じサービス・サブクラスに自動的にマップされるのではなく、それら自体のアクティビティー・タイプに従って処理されるようになります。例えば、CALL ステートメントがサービス・サブクラス SUBSC1 にマップされ、このルーチン内にオープン・カーソルがあるとします。このオープン・カーソルが、別のマッピング・ワーク・アクションの適用される別のワーク・クラスに属する場合、このオープン・カーソルは別のサービス・サブクラスへマップされます。

- ワーク・アクション (つまり、マッピング、しきい値、実行の阻止、アクティビティーのカウント、収集のアクション) に指定されているアクション・タイプを変更することができます。ただし、有効な作業タイプに変更しなければなりません。例えば、ワーク・アクションがアクティビティーのサービス・サブクラスへのマッピングである場合、このワーク・アクションをしきい値に変更したり、その逆を行ったりすることはできません。この理由は、この例の場合、マッピング・アクションのためにワーク・アクション・セットがサービス・スーパークラスに適用されていなければなりませんが、しきい値アクションは、サービス・スーパークラスに適用されるワーク・アクション・セットでは有効でないからです。しきい値ワーク・アクションであるワーク・アクションのタイプを変更した場合、またはワーク・アクションのタイプをしきい値に変更した場合、以下のことが生じます。
  - ワーク・アクションがしきい値で、非しきい値に変更された場合、そのしきい値は SYSCAT.THRESHOLDS ビューから除去されます。
  - ワーク・アクションがしきい値ではなく、しきい値に変更された場合、新規しきい値が SYSCAT.THRESHOLDS ビューに作成されます。

  注: アクションがしきい値である場合、そのしきい値のタイプを異なるしきい値に変更することはできません。したがって、例えばワーク・アクションが SQLROWSRETURNED しきい値であった場合、これを SQLTEMPSPACE しきい値に変更することはできません。さらに、有効にされた CONCURRENTDBCOORDACTIVITIES ワーク・アクションしきい値のワーク・アクション・タイプを変更することもできません。
- COLLECT AGGREGATE ACTIVITY DATA ワーク・アクションによって使用されるヒストグラム・テンプレートを変更して、対応するワーク・クラスに作成されるヒストグラムを示すことができます。ワーク・アクションによって使用されるヒストグラム・テンプレートを更新すると、SYSCAT.HISTOGRAMTEMPLATEUSE ビュー (そのサービス・クラスまたはワーク・アクションによって参照されるヒストグラム・テンプレートを表示する) 内の対応する行も更新されます。ヒストグラムおよびヒストグラム・テンプレートについて詳しくは、 140 ページの『ワークロード管理のヒストグラム』を参照してください。
- ワーク・アクションを使用可能にするか使用不可にするか。デフォルトでは、ワーク・アクションは使用可能になっています。使用可能に設定された場合、データ・サーバーは、このワーク・アクションのワーク・クラスに分類される

アクティビティーに対して、このワーク・アクションを適用することを考慮します。ワーク・アクションを使用不可にすると、そのワーク・アクションはデータ・サーバーに無視されます。

2. 変更をコミットします。 変更をコミットすると、ワーク・アクションは SYSCAT.WORKACTIONS ビューで更新されます。

ワーク・アクションを使用不可にする

ワーク・クラスに適用されないように、ワーク・アクションを使用不可にすることができます。実行時に、使用不可にされたワーク・アクションは、まるで存在しないかのように扱われます。

ワーク・アクションを使用不可にするためには、SYSADM または DBADM 権限が必要です。

ワーク・アクションを使用不可にするには、次のようにします。

1. ワーク・アクション・セットを作成するのか変更するのかに応じて、以下のステートメントのいずれかを使用します。
   - CREATE WORK ACTION SET ステートメントの DISABLE キーワードおよび ADD キーワードを使用します。以下に例を示します。

     ```
     ADD WORK ACTION work-action-name ON WORK CLASS work-class-name ... DISABLE
     ```

   - ALTER WORK ACTION SET ステートメントの DISABLE キーワードおよび ALTER キーワードを使用します。以下に例を示します。

     ```
     ALTER WORK ACTION work-action-name ... DISABLE
     ```

2. 変更をコミットします。 変更をコミットすると、ワーク・アクションは SYSCAT.WORKACTIONS ビューで更新されます。

ワーク・アクションのドロップ

必要でなくなったワーク・アクションは、ワーク・アクション・セットからドロップすることができます。

- ワーク・アクションをドロップするためには、SYSADM または DBADM 権限が必要です。
- その他の前提条件は、313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』を参照してください。

ワーク・アクションをドロップするには、次のようにします。

1. ALTER WORK ACTION SET ステートメントの DROP キーワードを使用します。 CONCURRENTDBCOORDACTIVITIES しきい値ワーク・アクションをドロップする場合は、1 回目の ALTER WORK ACTION SET 操作でそのワーク・アクションを使用不可に設定して変更をコミットし、それから 2 回目の ALTER WORK ACTION SET 操作でしきい値をドロップしてください。
2. 変更をコミットします。 変更をコミットすると、ワーク・アクションは SYSCAT.WORKACTIONS ビューから除去されます。ワーク・アクションがしきい値ワーク・アクションである場合、しきい値も SYSCAT.THRESHOLDS ビューから除去されます。

変更されたワーク・アクション・セットおよびワーク・アクションは、コミットされた後にはじめてデータベースで有効になり、現在実行中のデータベース・アクティビティーには影響しません。

ワーク・クラス・セットおよびワーク・クラスの作業

ワーク・クラス・セットの作成

ワーク・クラス・セットを作成するには、CREATE WORK CLASS SET ステートメントを使用します。

ワーク・クラス・セットを作成するためには、SYSADM または DBADM 権限が必要です。

その他の前提条件については、以下のトピックを参照してください。

- 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』
- 命名規則

ワーク・クラス・セットを作成するには、次のようにします。

1. CREATE WORK CLASS SET ステートメントを使用して、以下に挙げるワーク・クラス・セットのプロパティーを指定します。
   - ワーク・クラス・セットの名前。指定する名前は、データベースの中で固有でなければなりません。
   - オプション: ワーク・クラス・セットの 1 つ以上のワーク・クラス。詳しくは、 120 ページの『ワーク・クラスの作成』を参照してください。
2. 変更をコミットします。 変更をコミットすると、ワーク・クラス・セットが SYSCAT.WORKCLASSSETS ビューに追加されます。

ワーク・クラス・セットの変更

ワーク・クラス・セットの属性をワーク・クラス・セットの作成後に変更することはできません。ただし、ALTER WORK CLASS SET ステートメントを使用して、ワーク・クラス・セットに含まれるワーク・クラスを追加、変更、およびドロップすることはできます。

ワーク・クラス・セットを変更するためには、SYSADM または DBADM 権限が必要です。

その他の前提条件については、以下のトピックを参照してください。

- 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』
- 命名規則

1. ワーク・クラス・セットにワーク・クラスを追加する場合は、ADD キーワードを使用します。 ワーク・クラスを追加するときに指定できる各キーワードについては、 120 ページの『ワーク・クラスの作成』を参照してください。
2. ワーク・クラスを変更する場合は、ALTER キーワードを使用します。 ワーク・クラスを変更する方法については、 123 ページの『ワーク・クラスの変更』を参照してください。

3. ワーク・クラスをドロップする場合は、DROP キーワードを使用します。 ワーク・クラス・セットからワーク・クラスをドロップする方法については、 124 ページの『ワーク・クラスのドロップ』を参照してください。ワーク・クラス・セットからすべてのワーク・クラスをドロップする場合は、そのワーク・クラス・セット自体をドロップすることができます。詳しくは、『ワーク・クラス・セットのドロップ』を参照してください。
4. 変更をコミットします。 変更をコミットすると、SYSCAT.WORKCLASSES ビューが更新されて、追加、変更、またはドロップされたワーク・クラスがあればそれらが示されます。

ワーク・クラス・セットのドロップ

ワーク・クラス・セットをドロップするには、DROP WORK CLASS SET ステートメントを使用します。

ワーク・クラス・セットをドロップするためには、SYSADM または DBADM 権限が必要です。

ワーク・クラス・セットに関連付けられているワーク・アクション・セットがない場合にのみ、ワーク・クラス・セットをドロップできます。ワーク・クラス・セットをドロップする場合は、まずそれに従属するワーク・アクション・セットをドロップする必要があります。

ワーク・クラス・セットをドロップするには、次のようにします。

1. DROP WORK CLASS SET ステートメントを使用します。
2. 変更をコミットします。 変更をコミットすると、ワーク・クラス・セットが SYSCAT.WORKCLASSSETS ビューから除去されます。さらに、そのワーク・クラス・セットの一部だったすべてのワーク・クラスが、SYSCAT.WORKCLASSES ビューから除去されます。

ワーク・クラスの作成

ワーク・クラスを作成するには、CREATE WORK CLASS SET ステートメントまたは ALTER WORK CLASS SET ステートメントを使用します。

ワーク・クラスを作成するためには、SYSADM または DBADM 権限が必要です。

その他の前提条件については、以下のトピックを参照してください。

- 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』
- 命名規則

ワーク・クラスを作成するには、次のようにします。

1. ワーク・クラスを作成して同時に新規ワーク・クラス・セットも作成する場合、または新規ワーク・クラスを既存のワーク・クラス・セットに追加する場合は、以下のようにします。
   - 新規ワーク・クラス・セットに追加する新規ワーク・クラスを作成するには、CREATE WORK CLASS SET ステートメントの WORK CLASS キーワードを使用します。

- 既存のワーク・クラス・セットに追加する新規ワーク・クラスを作成するには、ALTER WORK CLASS SET ステートメントの ADD WORK CLASS キーワードを使用します。

新規ワーク・クラスに、以下に挙げるプロパティーを 1 つ以上指定します。

- ワーク・クラスの名前。この名前は、ワーク・クラス・セット内で固有でなければなりません。
- ワーク・クラスの属性。これらの属性は、アクティビティーをワーク・クラスと関連付けるために使用されます。
  - ワーク・クラスを使用する作業のタイプ。この特性を指定するには WORK TYPE パラメーターを使用します。
    - READ。非更新 SELECT アクティビティー、およびすべての XQuery アクティビティーを表します。

      READ キーワードを指定するときには、オプションの for-from-to-clause 引数も指定できます。この引数を使用して、ステートメントのコスト (timeron 単位で) またはステートメントのカーディナリティー (戻される行数) の範囲を指定します。最初の値には数値を指定する必要があります。2 番目の値には、数値を指定するか、または値 UNBOUNDED を指定してアクティビティーのコストまたはカーディナリティーに上限を設けないことを指定できます。この引数は、WRITE キーワード、DML キーワード、および ALL キーワードに対しても指定できます。

      例えば、5000 timeron 以上のコストの SELECT アクティビティーをこのワーク・クラスに関連付けるには、以下のように指定します。

      ```
      WORK TYPE READ FOR TIMERONCOST FROM 5000 TO UNBOUNDED
      ```

    - WRITE。データベース内のデータを更新する SQL アクティビティーを表します。

      例えば、50 から 100 行を更新するデータ書き込みアクティビティーをこのワーク・クラスに関連付けるには、以下のように指定します。

      ```
      WORK TYPE WRITE FROM 50 TO 100
      ```

    - CALL。CALL アクティビティーを表します。

      CALL キーワードを指定するときに、ROUTINES IN SCHEMA キーワードも指定して、特定のスキーマ内のルーチンへの CALL アクティビティーのみがこのワーク・クラスに関連付けられるようにすることができます。例えば、ACCOUNTS スキーマ内のルーチンへの呼び出しのみをこのワーク・クラスに関連付ける場合、以下のように指定します。

      ```
      WORK TYPE CALL ROUTINES IN SCHEMA ACCOUNTS
      ```

    - DML。READ および WRITE キーワードの両方によってカバーされる SQL アクティビティーを表します。

      例えば、500 から 1000 timeron のコストの DML アクティビティーすべてをこのワーク・クラスに関連付けるには、以下のように指定します。

      ```
      WORK TYPE DML FOR TIMERONCOST FROM 500 TO 1000
      ```

    - DDL。以下のアクティビティーを表します。
      - ALTER

- CREATE
- COMMENT
- DECLARE GLOBAL TEMPORARY TABLE
- DROP
- FLUSH PACKAGE CACHE
- GRANT
- REFRESH TABLE
- RENAME
- REVOKE
- SET INTEGRITY

例えば、すべての DDL アクティビティーをこのワーク・クラスに関連付けるには、以下のように指定します。

```
WORK TYPE DDL
```

- LOAD。LOAD アクティビティーを表します。

例えば、LOAD アクティビティーをこのワーク・クラスに関連付けるには、以下のように指定します。

```
WORK TYPE LOAD
```

- ALL。先行するキーワードすべてによって示されるすべての作業タイプを表します。

ワーク・クラス・タイプに ALL を指定するときに、ROUTINES IN SCHEMA キーワードも指定して、特定のスキーマ内のルーチンへの CALL アクティビティーのみがこのワーク・クラスに関連付けられるようにすることができます。for-from-to-clause 引数を指定して、指定されている見積コスト (timeron) またはカーディナリティーの DML アクティビティーすべてをこのクラスに入れるようにすることもできます。例えば、300 から 1500 行のカーディナリティーを持つ DML アクティビティーと、NEWHIRES スキーマから呼び出されるルーチンの両方をこのワーク・クラスに関連付けるには、以下のように指定します。

```
WORK TYPE ALL FOR CARDINALITY FROM 300 TO 1500 ROUTINES
IN SCHEMA NEWHIRES
```

このワーク・クラスのタイプは ALL であるため、これは、LOAD アクティビティーおよび DDL アクティビティーといった、スキーマまたはカーディナリティーを持たない他のアクティビティーにも適用されます。

– オプション。ワーク・クラス・セット内でのワーク・クラスの位置。ワーク・クラス・セット内でのワーク・クラスの位置によって、アクティビティーをワーク・クラスにクラス分けする際にワーク・クラスが評価される順序が決定されます。ワーク・クラスの割り当てが行われるとき、データ・サーバーはまずオブジェクト (サービス・スーパークラスまたはデータベースのいずれか) に関連するワーク・クラス・セットを判別し、それからそのワーク・クラス・セット内で、ワーク・アクションが関連付けされた

最初に一致するワーク・クラスを選択します。POSITION キーワードを使用して、以下のいずれかを指定します。

- LAST。ワーク・クラスは、ワーク・クラス・セット内でワーク・クラスのリストの最後尾に配置されます。以下に例を示します。

  ```
  WORK TYPE ... POSITION LAST
  ```

- BEFORE *work-class-name*。ワーク・クラスは、ワーク・クラス・セット内の、指定のワーク・クラスの前に作成されます。以下に例を示します。

  ```
  WORK TYPE ... POSITION BEFORE LARGEDDL
  ```

- AFTER *work-class-name*。ワーク・クラスは、ワーク・クラス・セット内の、指定のワーク・クラスの後に作成されます。以下に例を示します。

  ```
  WORK TYPE ... POSITION AFTER LARGEDDL
  ```

- AT *integer*。ワーク・クラスは、ワーク・クラス・セット内の、整数値で指定された位置に作成されます。以下に例を示します。

  ```
  WORK TYPE ... POSITION AT 3
  ```

2. 変更をコミットします。 変更をコミットすると、ワーク・クラスは SYSCAT.WORKCLASSES ビューに追加されます。

ワーク・クラスの変更

ワーク・クラスを変更する必要がある場合には、ALTER WORK CLASS SET ステートメントを使用します。

ワーク・クラスを変更するためには、SYSADM または DBADM 権限が必要です。

その他の前提条件は、313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』を参照してください。

ワーク・クラスを変更するには、次のようにします。

1. ALTER WORK CLASS SET ステートメントの ALTER キーワードを使用して、以下のプロパティー (複数可) を変更します。 これらのプロパティーについてサポートされる値の説明は、120 ページの『ワーク・クラスの作成』を参照してください。
   - FOR キーワード。例えば、FOR キーワードに指定される値を CARDINALITY から TIMERONCOST に変更できます。
   - FROM *from-value* TO *to-value* 引数。例えば、引数を `FROM 50 TO 100` から `FROM 500 TO 1500` に変更できます。
   - CALL アクティビティーの SCHEMA キーワード。例えば、現在ワーク・クラスでスキーマの指定がない場合、スキーマを追加できます。また、キーワード ALL を指定して、ルーチンのスキーマに関わりなく、ワーク・クラスがすべての CALL ステートメントに適用されるようにすることもできます。ALL はデフォルトです。
   - POSITION キーワード。例えば、AT キーワードを使用して、最後の位置から任意の位置に、または LAST キーワードを使用して任意の位置から最後の位置に、ワーク・クラスを移動させることができます。
2. 変更をコミットします。 変更をコミットすると、ワーク・クラスは SYSCAT.WORKCLASSES ビューで更新されます。

ワーク・クラスのドロップ

必要でなくなったワーク・クラスは、ワーク・クラス・セットからドロップすることができます。

ワーク・クラスをドロップするためには、SYSADM または DBADM 権限が必要です。

その他の前提条件は、 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』を参照してください。

ワーク・クラスをドロップするには、次のようにします。

1. ALTER WORK CLASS SET ステートメントの DROP キーワードを使用します。 ワーク・クラス・セットと関連付けられているワーク・アクション・セット内に、ドロップしたいワーク・クラスに依存しているワーク・アクションがある場合には、そのワーク・クラスをドロップすることはできません。この場合、まず従属ワーク・アクションすべてをドロップしてから、ワーク・クラスをドロップする必要があります。
2. 変更をコミットします。 変更をコミットすると、ワーク・クラスは SYSCAT.WORKCLASSES ビューから除去されます。

第 3 部 モニターおよび制御

© Copyright IBM Corp. 2007, 2008

第 6 章 モニターおよび制御

ワークロード管理フィーチャーおよび機能により、データベース内で実行中の作業のモニターと制御の両方を行うことができます。

例えば、以下のタスクを実行できます。

- 初期ワークロード管理構成の設計を支援するための、システム上のワークロードの分析。
- 以下のタイプの操作情報を入手することによる、システムの動作のトラッキングおよび調査。
  - 環境に関する一般モニター情報。
  - システム・パフォーマンス低下を分析するための情報。
  - 停止アクティビティーを診断するための情報。
  - エージェント競合を調査するための情報。
  - パフォーマンスが悪い照会を分離するための情報。

  アクティビティー、サービス・クラス、ワークロード、ワーク・クラス、しきい値キュー、およびしきい値違反についての情報を入手できます。
- 問題を引き起こすと予想される、キューに入れられたアクティビティーを取り消すことで環境を制御します。またはシステムにマイナスの影響を与えると診断されたアクティビティーの実行を取り消します。

ワークロード管理ソリューションでは、データベースからサービス・クラスとワークロードまで、およびデータベース内の個々のアクティビティーまでドリルダウンできるので、より効率的にトラブルシューティングを行うことができます。

データのモニターの概要

データのモニターは、ワークロード、ワーク・クラス、サービス・サブクラス、サービス・スーパークラス、およびしきい値キューから使用可能です。このデータを使用して、問題を診断および修正したり、パフォーマンス調整を行ったりすることができます。

以下の図は、ワークロードで使用可能なモニター情報を示しています。イベント・モニターを使用して、ワークロード内で実行されるアクティビティーに関するワークロード統計および情報を収集することができます。表関数を使用して、ワークロード統計およびワークロード・オカレンスに関する情報にリアルタイムでアクセスすることができます。

© IBM Corporation 2007, 2008

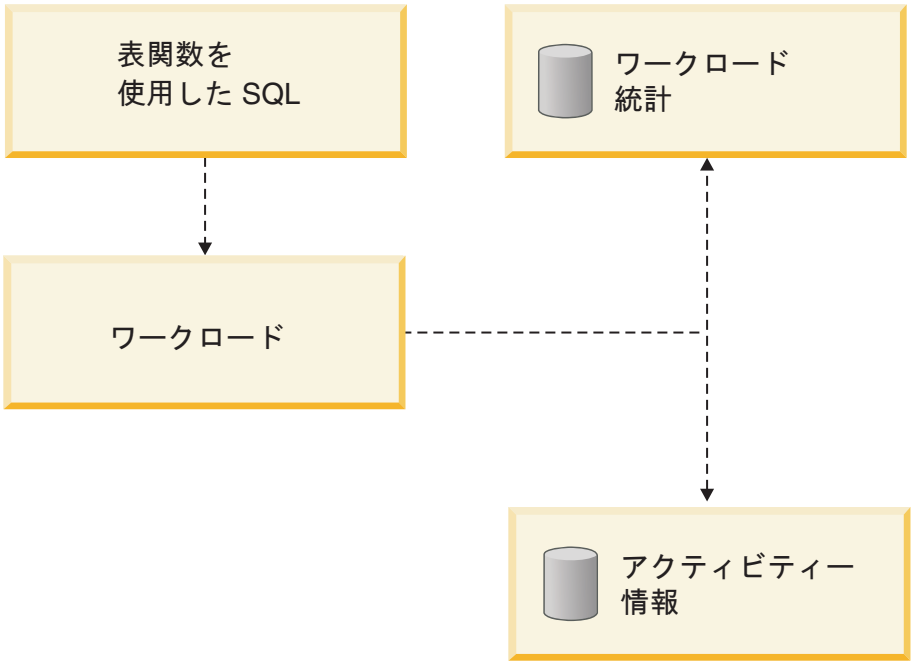


図 20. ワークロードで使用可能なデータのモニター

以下の図は、サービス・クラスで使用可能なモニター情報を示しています。サービス・サブクラスおよびサービス・スーパークラスの統計を収集することができます。サービス・サブクラスの場合、集約アクティビティーおよび要求統計、およびサービス・サブクラスで実行されるアクティビティーに関する情報を入手することもできます。表関数を使用して、サービス・スーパークラスおよびサービス・サブクラスの統計、および特定のサービス・クラスで実行中のエージェントに関する情報にリアルタイムでアクセスできます。

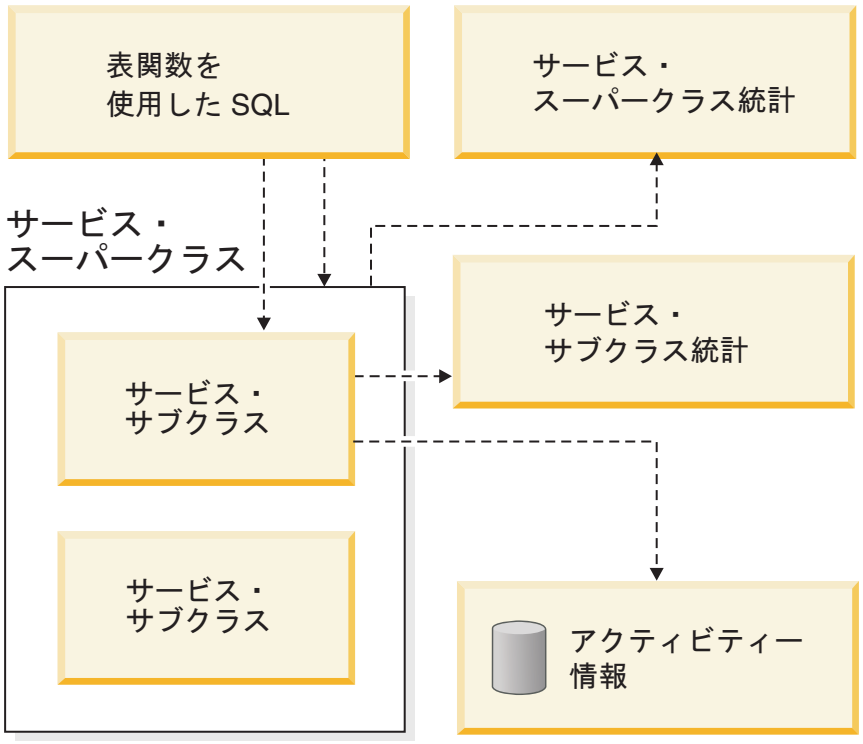


図 21. サービス・クラスで使用可能なデータのモニター

以下の図は、ワーク・クラスで使用可能なモニター情報を示しています。ワーク・クラスの統計、および特定のワーク・クラスに関連付けられたアクティビティーに関する情報を収集することができます。表関数を使用して、ワーク・クラスの統計にリアルタイムでアクセスすることができます。

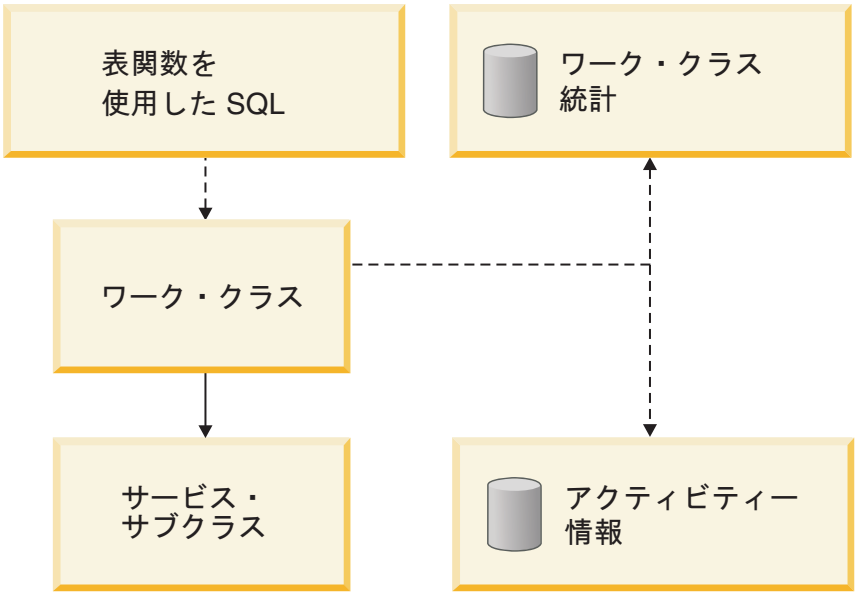


図 22. ワーク・クラスで使用可能なデータのモニター

以下の図は、しきい値で使用可能なモニター情報を示しています。しきい値の違反、しきい値の違反の原因となったアクティビティー、およびキューイング統計

(キューイングしきい値用) に関する情報を入手できます。表関数を使用して、キューイングしきい値の統計にリアルタイムでアクセスすることができます。

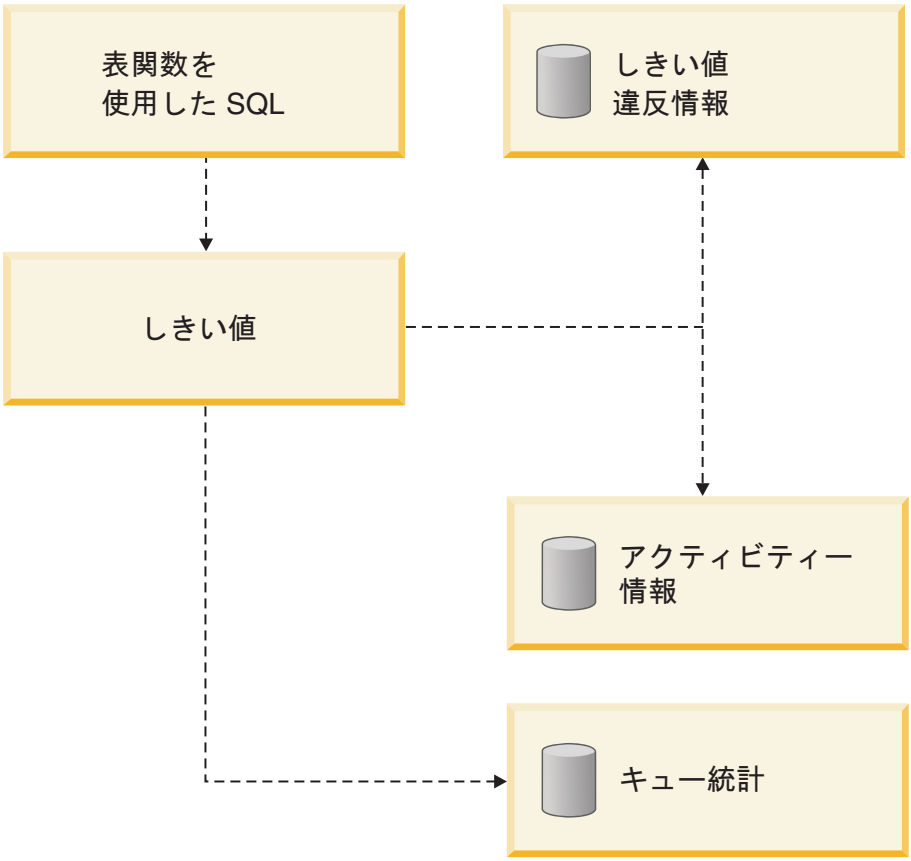


図 23. しきい値で使用可能なデータのモニター

操作情報を入手するためのワークロード管理の表関数

このトピックで説明されている表関数を使用して、操作情報を入手することができます。

ワークロード管理の表関数は SYSPROC スキーマに用意されています。これらの表関数はハイパフォーマンスであり、現在実行中のワークロードにほとんど影響を与えることなく、システムで実行されている作業に関する情報を返すことができます。

以下の表関数を使用して、システム上で実行されている作業について、サービス・クラス、ワークロード・オカレンス、エージェント、要求、およびアクティビティーの観点から調べることができます。すべての表関数は、パーティション・データベース環境の単一データベース・パーティションまたはすべてのデータベース・パーティションのいずれかに関する情報を返すことができます。

- WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES (*service_superclass_name*, *service_subclass_name*, *dbpartitionnum*)。この表関数は、データベース内のワークロード・オカレンスのリストを入手するために使用しま

す。ワークロード・オカレンスは、ワークロード定義と一致する属性を持つデータベース接続です。特定のサービス・クラスあるいはすべてのサービス・クラスのワークロード・オカレンスをリストすることができます。この表関数の使用についての詳しい情報は、 199 ページの『例: サービス・クラスによるエージェント使用の調査』を参照してください。

- WLM_GET_SERVICE_CLASS_AGENTS(*service_superclass_name*, *service_subclass_name*, *app_handle*, *dbpartitionnum*)。この表関数は、データベース内で作業中のエージェントのリストを入手するために使用します。特定のサービス・クラスで実行中のすべてのエージェント、または特定のアプリケーションのために作業中のすべてのエージェントをリストすることができます。さらに、この表関数を使用して、アプリケーションのコーディネーター・エージェントおよびサブエージェントの状態を判別したり、システム内の各エージェントが作業中の要求を判別したりすることができます。この表関数の使用についての詳しい情報は、 199 ページの『例: サービス・クラスによるエージェント使用の調査』を参照してください。
- WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(*app_handle*, *dbpartitionnum*)。この表関数は、特定のデータベース・パーティションのワークロード・オカレンスに関連付けられたアクティブなアクティビティーのリストを入手するために使用します (複数のアプリケーション ID およびデータベース・パーティションにまたがる場合は、ワイルドカード文字を使用できます)。この表関数は、キューに入れられたアクティビティー、アイドル状態のアクティビティー、または実行中のアクティビティーに関する情報を返します。この表関数は、実行が完了したアクティビティーに関する情報は返しません。この表関数の使用についての詳しい情報は、 187 ページの『例: ワークロード管理の表関数を使用したデータの集約』および 194 ページの『例: ハング・アクティビティーの識別』を参照してください。
- WLM_GET_ACTIVITY_DETAILS(*app_handle*, *uow_id*, *activity_id*, *dbpartitionnum*)。この表関数は、進行中のアクティビティーに関する詳細情報を入手するために使用します。アクティビティーの識別は、アクティビティー ID、作業単位 ID、およびアプリケーション ID の固有の組み合わせによって行います。この表関数を使用して、WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数によって返されるアクティビティーに関する情報を分析することができます。この表関数の使用についての詳しい情報は、 183 ページの『例: ワークロード管理の表関数を使用して、異なるレベルで現行システムの動作をモニターする』を参照してください。

ワークロード管理表関数とスナップショット・モニターの統合

問題判別またはパフォーマンス調整を実行する際、ワークロード管理表関数をスナップショット・モニター表関数とともに使用できます。

ワークロード管理表関数とスナップショット・モニター表関数は、以下のフィールドを共有します。こうしたフィールドでデータを結合して、診断およびパフォーマンス調整を実施するのに必要なデータを得ることができます。

表 15. ワークロード管理表関数とスナップショット・モニター表関数で共有するフィールド

ワークロード・マネージャー表関数フィールド	スナップショット・モニター表関数フィールド
agent_tid	**agent_pid**
application_handle	**agent_id** **agent_id_holding_lock**
session_auth_id	**session_auth_id**
dbpartitionnum	**node_number**
utility_id	**utility_id**
workload_id	**workload_id**

異なる表関数の間で結合を使用する理由を示す例として、BATCH サービス・スーパークラスで実行されているすべてのユーティリティーに関する基本情報を取得したいとします。以下の照会を発行できます。

```
SELECT SUBSTR(UTILITY_TYPE,1,4) TYPE,
       UTILITY_PRIORITY PRIORITY,
       SUBSTR(UTILITY_DESCRIPTION,1,12) AS UTILITY_DESCRIPTION,
       SUBSTR(UTILITY_DBNAME,1,8) AS DBNAME,
       UTILITY_STATE,
       SUBSTR(UTILITY_INVOKER_TYPE,1,7) INVOKER,
       SUBSTR(CHAR(WLM.DBPARTITIONNUM),1,4) PART,
       SUBSTR(CLASSES.PARENTSERVICECLASSNAME,1,19) SUPERCLASS_NAME,
       SUBSTR(CLASSES.SERVICECLASSNAME,1,18) SUBCLASS_NAME
FROM SYSIBMADM.SNAPUTIL SNAP,
     TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(CAST(NULL AS BIGINT), -2)) WLM,
     SYSCAT.SERVICECLASSES CLASSES
WHERE SNAP.UTILITY_ID = WLM.UTILITY_ID
  AND WLM.SERVICE_CLASS_ID = CLASSES.SERVICECLASSID
  AND CLASSES.SERVICECLASSNAME = 'SYSDEFAULTSUBCLASS'
  AND CLASSES.PARENTSERVICECLASSNAME = 'BATCH'
ORDER BY WLM.DBPARTITIONNUM;
```

出力は、以下のようになります。

```
TYPE PRIORITY    UTILITY_DESCRIPTION DBNAME   UTILITY_STATE INVOKER PART SUPERCLASS_NAME     SUBCLASS_NAME
---- ----------- ------------------- -------- ------------- ------- ---- ------------------- ------------------
LOAD           - OFFLINE LOAD        SAMPLE   EXECUTE       USER    1    BATCH               SYSDEFAULTSUBCLASS
LOAD           - OFFLINE LOAD        SAMPLE   EXECUTE       USER    1    BATCH               SYSDEFAULTSUBCLASS
LOAD           - OFFLINE LOAD        SAMPLE   EXECUTE       USER    1    BATCH               SYSDEFAULTSUBCLASS
LOAD           - OFFLINE LOAD        SAMPLE   EXECUTE       USER    2    BATCH               SYSDEFAULTSUBCLASS
LOAD           - OFFLINE LOAD        SAMPLE   EXECUTE       USER    2    BATCH               SYSDEFAULTSUBCLASS
LOAD           - OFFLINE LOAD        SAMPLE   EXECUTE       USER    2    BATCH               SYSDEFAULTSUBCLASS
LOAD           - OFFLINE LOAD        SAMPLE   EXECUTE       USER    3    BATCH               SYSDEFAULTSUBCLASS
LOAD           - OFFLINE LOAD        SAMPLE   EXECUTE       USER    3    BATCH               SYSDEFAULTSUBCLASS
LOAD           - OFFLINE LOAD        SAMPLE   EXECUTE       USER    3    BATCH               SYSDEFAULTSUBCLASS
```

ワークロード管理ストアード・プロシージャー

アクティビティーをキャンセルしたり、アクティビティーに関する詳細情報をキャプチャーしたり、ワークロード管理オブジェクトに関する統計をリセットしたりするために、ストアード・プロシージャーを使用することができます。

以下のストアード・プロシージャーを使用できます。

- WLM_CANCEL_ACTIVITY(*application_handle*, *uow_id*, *activity_id*)。このストアード・プロシージャーは、実行中またはキューに入れられたアクティビティーをキャンセルするために使用します。アクティビティーは、そのアプリケーション・

ハンドル、作業単位 ID、およびアクティビティー ID で識別されます。どのタイプのアクティビティーもキャンセルすることができます。キャンセルされたアクティビティーが含まれるアプリケーションはエラー SQL4725N を受け取ります。

- WLM_CAPTURE_ACTIVITY_IN_PROGRESS(*application_handle*, *uow_id*, *activity_id*)。このストアード・プロシージャーを使用して、現在実行中の個別のアクティビティーに関する情報をアクティビティー・イベント・モニターに送信します。このストアード・プロシージャーは、アクティビティーが完了するまで待機するのではなく、情報を即時に送信します。

  注: このストアード・プロシージャーを使用して、INOUT パラメーターが含まれるプロシージャーのアクティビティー情報を収集する場合、INOUT 値はキャプチャーが行われるまでに上書きされる可能性があります。アクティビティー・データをキャプチャーするサービス・クラス、ワークロード、ワーク・アクション、または予測しきい値を COLLECT ACTIVITY WITH DETAILS AND VALUES として作成した場合、あるいはサービス・クラス、ワークロード、ワーク・アクション、または予測しきい値を変更して、ON COORDINATOR または ON ALL キーワードのいずれか、および WITH DETAILS AND VALUES キーワードとともに COLLECT ACTIVITY DATA キーワードを指定した場合、この状態は発生しません。
- WLM_COLLECT_STATS()。このストアード・プロシージャーは、ワークロード管理オブジェクトの統計を収集およびリセットするために使用します。サービス・クラス、ワークロード、しきい値キュー、およびワーク・アクション・セット用に追跡されたすべての統計は、アクティブな統計イベント・モニターに送信され (存在する場合)、リセットされます。アクティブな統計イベント・モニターが存在しない場合、統計のリセットのみが行われ、収集は行われません。

ワークロード管理イベント・モニター

イベント・モニターは、デバッグ用に履歴情報を収集したり、イベントのセットをキャプチャーしたりします。一方、表関数は特定時点の情報を収集および報告します。

ワークロード管理構成において、以下のタイプのイベント・モニターを使用することができます。

- ACTIVITIES。このタイプのイベント・モニターは、個別のアクティビティーに関する情報をキャプチャーします。場合によっては、ステートメント情報、および SQL アクティビティーの変数に対する入力データをイベント・モニターに含めることができます。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は、db2advis などのツールへの入力として使用することができます。さらに、ACTIVITIES イベント・モニターを使用して、個々のアクティビティーをデバッグするための情報をキャプチャーすることもできます。

  アクティビティーに関する情報を収集するには、該当するアクティビティーが属するサービス・クラス、ワークロード、またはワーク・アクションに対して、あるいは該当するアクティビティーが違反する可能性のあるしきい値に対して、COLLECT ACTIVITY DATA を指定します。アクティビティーが完了すると、アクティビティーが正常に完了したかどうかにかかわらず、情報が収集されます。

- THRESHOLD VIOLATIONS。このタイプのイベント・モニターは、アクティビティーがしきい値に違反するたびに情報をキャプチャーします。この情報に組み込まれるのは、しきい値に違反したアクティビティーを一意的に識別するための ID、作業単位、およびアプリケーション・ハンドル、ならびにアクティビティーに適用されるアクション (STOP EXECUTION または CONTINUE) です。

  しきい値に COLLECT ACTIVITY DATA を指定した場合、アクティビティー・イベント・モニターが作成されてアクティブになると、しきい値に違反しているアクティビティーに関する情報も収集されますが、この情報が収集されるのはアクティビティーが終了 (正常に完了あるいは失敗) したときです。

  しきい値に関する詳細を入手するには SYSCAT.THRESHOLDS ビューを照会します。
- STATISTICS。このタイプのイベント・モニターは、設定した一定期間に渡って測定される統計をキャプチャーします。 STATISTICS イベント・モニターは、個別のアクティビティーではなく集約アクティビティー情報を処理し、そのターゲットを単一のサービス・クラスまたはワーク・クラスとすることができます。そのため、このタイプのイベント・モニターは、ステートメントまたはアクティビティー・イベント・モニターと比べて、履歴情報をキャプチャーするためのコストが低い方法です。イベント・モニターに統計を送信する方法について詳しくは、144 ページの『統計イベント・モニターを使用したワークロード管理統計の収集』を参照してください。

ステートメント、接続、およびトランザクション・イベント・モニターとは異なり、アクティビティー、統計、およびしきい値違反のイベント・モニターにはイベント条件 (つまり、CREATE EVENT MONITOR ステートメントの WHERE キーワードで指定された条件) がありません。代わりに、これらのイベント・モニターは、サービス・クラス、ワークロード、ワーク・クラス、およびしきい値の属性に依存することによって、これらのオブジェクトが自分のアクティビティー情報または集約情報を各モニターに送るかどうかを決めます。

通常、イベント・モニターは表またはファイルのいずれかにデータを書き込みます。これらの表またはファイルから自動的にデータが削除されるわけではないため、自分で定期的にデータを削除する必要があります。

sqllib/misc ディレクトリーの wlmevmon.ddl スクリプトを使用して、DB2ACTIVITIES、DB2STATISTICS、および DB2THRESHOLDVIOLATIONS という 3 つのイベント・モニターを作成して使用可能にすることができます。必要に応じて、スクリプトを変更して表スペースまたは他のパラメーターを変更してください。

統計の管理

ワークロード管理オブジェクトの統計

サービス・クラス、ワーク・クラス、ワークロード、およびしきい値キューを含む、ワークロード管理オブジェクトの統計が維持されます。これらの統計はメモリー内に常駐しており、ワークロード管理統計の表関数を使用してリアルタイムで表示することが可能です。あるいは、統計を収集して統計イベント・モニターに送信し、後で履歴分析を行うときに表示することもできます。

統計がイベント・モニターに送信されると、メモリー内の値はリセットされ、重複したデータが後続の収集間隔において収集されないようにします。ワークロード管理統計の表関数は現在のメモリー内の値を報告するため、収集の後はリセットされた値が報告されます。ワークロード管理の表関数は統計のサブセットのみを報告します。完全な統計のセットを表示するには、統計を収集して、それらを統計イベント・モニターに送信する必要があります。

以下の統計は、各データベース・パーティションの所定のオブジェクトに関して維持されます。これは、そのオブジェクトを作成または変更したときに指定した COLLECT AGGREGATE ACTIVITY DATA または COLLECT AGGREGATE REQUEST DATA オプションの値に関わりなく、そのようになります。

- しきい値キュー:
  - キューの割り当ての合計 (**queue_assignments_total**)。この統計は、過剰なキューイングが発生していないかどうか、あるいは適正な数のアクティビティーがキューイングされているかどうか (つまり、並行性しきい値による制限が過度または不十分ではないか) を判別するために使用します。
  - キュー・サイズの上限 (**queue_size_top**)。この統計は、最大キュー・サイズを決定したり、キュー・サイズが十分であるかどうかを識別したりするために使用します。
  - キュー時間の合計 (**queue_time_total**)。この統計は、アクティビティーがキューで費やす時間や、その時間が過剰でないかを判別するために使用します。
- サービス・サブクラス:
  - 並行アクティビティーの上限 (**concurrent_act_top**)。この統計は、統計を収集した時間間隔中に、特定のデータベース・パーティションで特定のサービス・クラスが達した、アクティビティーの最高の並行性 (ネストされたアクティビティーを含む) を判別するために使用します。
  - 完了したコーディネーター・アクティビティーの合計 (**coord_act_completed_total**)。この統計は、サービス・クラスで実行中の作業の量を判別するために使用します。
  - 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 (**coord_act_aborted_total**)。この統計は、正常に完了しなかったアクティビティーを測定し、システムがどれほど正常稼働しているかを判別するために使用します。アクティビティーは、取り消し、エラー、または再アクティブしきい値が原因で打ち切られる可能性があります。
  - 拒否されたコーディネーター・アクティビティーの合計 (**coord_act_rejected_total**)。この統計は、アクティビティーの拒否を測定し、拒否ポリシーの有効性の指標を入手するために使用します。アクティビティーが STOP EXECUTION アクションを持つ予測しきい値に違反した場合、あるいはワーク・アクションによって実行を妨げられた場合、そのアクティビティーは拒否されたものとして数えられます。
  - アクティブな要求の数 (**num_requests_active**)。この統計は、サービス・クラスで現在実行中の要求の数を判別するために使用します。
- サービス・スーパークラス:
  - 同時接続の上限 (**concurrent_connection_top**)。この統計は、接続の並行性しきい値を調整するために使用します。
- ワークロード:

- 完了したワークロード・オカレンスの合計 (**wlo_completed_total**)。この統計は、特定の期間に完了したワークロードのオカレンスの数を判別するために使用します。
- 並行ワークロード・オカレンスの上限 (**concurrent_wlo_top**)。この統計は、並行ワークロード・オカレンスの最大数を識別するために、あるいは並行して実行中のワークロード・オカレンス数が多すぎる (つまり、同じワークロード定義に関連付けられ、システム上で同時に実行しているアプリケーションが多すぎる) 場合にワークロード・オカレンスの並行性しきい値を設定または調整するために使用します。
- 並行アクティビティーの上限 (**concurrent_act_top**)。この統計は、CONCURRENTWORKLOADACTIVITIES しきい値を調整するために使用します。
- 完了したコーディネーター・アクティビティーの合計 (**coord_act_completed_total**)。この統計は、アクティビティーの正常完了率を測定し、システムの正常性の指標を入手するために使用します。
- 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 (**coord_act_aborted_total**)。この統計は、正常に完了しなかったアクティビティーを測定し、システムがどれほど正常稼働しているかを判別するために使用します。アクティビティーは、取り消し、エラー、または再アクティブしきい値が原因で打ち切られる可能性があります。
- 拒否されたコーディネーター・アクティビティーの合計 (**coord_act_rejected_total**)。この統計は、アクティビティーの拒否率を測定し、拒否ポリシーの有効性を判別するために使用します。アクティビティーが STOP EXECUTION アクションを持つ予測しきい値に違反した場合、あるいはワーク・アクションによって実行を妨げられた場合、そのアクティビティーは拒否されたものとして数えられます。
- 完了したワークロード・オカレンスの合計 (**wlo_completed_total**)。この統計は、特定の期間に完了したワークロードのオカレンスの数を判別するために使用します。

- ワーク・クラス (ワーク・アクションを介した):
  - アクティビティーの合計 (**act_total**)。この統計は、ワーク・アクション・セットの有効性を判別したり、システム上の各アクティビティー・タイプの相対的な割合を判別したりするために使用します。

サービス・サブクラスの COLLECT AGGREGATE REQUEST DATA オプションの値を BASE に設定すると、以下の統計がサービス・サブクラス用に維持されます。

- 要求実行時間のヒストグラム。要求実行時間は、データベース・パーティションごとにすべての要求に関して収集され、ヒストグラムにまとめられます。要求実行時間は、アクティビティーのために作業するエージェントが費やす時間を概算したものです (アクティビティーは 1 つ以上の要求で構成されます)。この情報は、どこで作業が実行されているか、およびパーティション間で作業の分散が均等であるかどうかを把握するために使用します。 (例えば、コーディネーター・アクティビティー・カウントは、大半のアクティビティーが 1 つのデータベース・パーティションで発生していることを示しているかもしれませんが、コーディネーター・エージェントは、アクティビティーの処理の一環として、要求を別のデータベース・パーティションに送り、そこで作業の大半が実行される場合が

あります。) 要求実行時間のヒストグラムは、データベース・パーティションに送信される要求のサイズを判別するのに役立ちます (つまり、データベース・パーティションに送信される作業の大半を構成しているのが小さな要求あるいは大きな要求のどちらか、または特定の分散がないかどうか)。

- 要求実行時間の平均 (**request_exec_time_avg**)。この統計は、データベース・パーティションでの各要求の処理に費やされる平均時間を把握し、対応する要求実行時間のヒストグラムのヒストグラム・テンプレートを調整するために使用します。

サービス・サブクラスまたはワーク・クラス (ワーク・アクションを介した) に対して COLLECT AGGREGATE ACTIVITY DATA オプションの値を BASE に設定すると、対応するサービス・クラスまたはワーク・クラスに関して、データベース・パーティションごとに、以下の統計が収集されるか、以下のヒストグラムが生成されます。平均を使用すると、アクティビティーがどこで時間の大半を費やすか (例えば、キューで、あるいは実行時に)、およびその応答時間 (存続時間) を素早く把握することができます。さらに、平均を使用してヒストグラム・テンプレートを調整することもできます。つまり、真の平均とヒストグラムから計算された平均とを比較して、ヒストグラムからの平均が真の平均から外れている場合、自分のデータにとってより適切なビン値のセットを使用して、対応するヒストグラムのヒストグラム・テンプレートを変更することを考慮できます。

- コーディネーター・アクティビティーの平均存続期間 (**coord_act_lifetime_avg**)。この統計は、サービス・クラスまたはワーク・クラスに関連付けられた、ネストなしコーディネーター・アクティビティーの存続期間の算術平均を判別するために使用します。
- コーディネーター・アクティビティーの平均実行時間 (**coord_act_exec_time_avg**)。この統計は、サービス・クラスまたはワーク・クラスに関連付けられた、ネストなしコーディネーター・アクティビティーの実行時間の算術平均を判別するために使用します。
- コーディネーター・アクティビティー・キューの平均時間 (**coord_act_queue_time_avg**)。この統計は、サービス・クラスまたはワーク・クラスに関連付けられた、ネストなしコーディネーター・アクティビティーのキュー時間の算術平均を判別するために使用します。
- コスト見積もりの上限 (**cost_estimate_top**)。この統計は、見積コストのしきい値を調整するために使用します。
- 返される見積行数の上限 (**rows_returned_top**)。この情報は、返される実際の行数のしきい値を調整するために使用します。
- TEMPORARY 表スペースの上限 (**temp_tablespace_top**)。この統計は、TEMPORARY 表スペース使用量のしきい値を調整するために使用します。この統計は、TEMPORARY 表スペース使用量のしきい値を定義した場合にのみモニターされます。
- アクティビティーの存続期間のヒストグラム。このヒストグラムは、ネストなしコーディネーター・アクティビティーのアクティビティー発生時刻から終了時刻までの時間を収集します。このヒストグラムは、システム・パフォーマンスの全体像を把握するために使用します。アクティビティーが、終了後もカーソルを開いたままにするルーチンの場合、存続期間ヒストグラムは、カーソルの存続期間を、カーソルの親であるルーチンの存続期間にカウントしません。

- アクティビティー実行時間のヒストグラム。このヒストグラムは、ネストなしコーディネーター・アクティビティーの実行時間を収集します。このヒストグラムは、実行時間に影響する、システムに対する変更の影響を測定するために使用します。実行時間は以下のように計算されます。
  - カーソルの場合、実行時間はオープン・カーソル要求、フェッチ、およびクローズ・カーソル要求を組み合わせた時間になります。カーソルがアイドル中の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の時間です。ルーチンの終了後もカーソルが開かれたままの場合、これらの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、アクティビティーの存続期間とアクティビティーがキューで費やした時間との差になります。
- アクティビティー・キュー時間のヒストグラム。このヒストグラムは、ネストなしコーディネーター・アクティビティーがキューで費やす時間を収集します。このヒストグラムを使用して、アクティビティーに対するキューイングしきい値の影響を測定します。

サービス・サブクラスまたはワーク・クラスに対して COLLECT AGGREGATE ACTIVITY DATA オプションの値を EXTENDED に設定すると、対応するサービス・クラスまたはワーク・クラス (ワーク・アクションを介した) に関して、データベース・パーティションごとに、以下のシステム統計が収集されるか、以下のヒストグラムが生成されます。平均を使用すると、アクティビティーの発生の平均比率 (発生比率は発生間隔時間の反対) およびアクティビティーのコスト (見積コスト) について素早く理解することができます。さらに、平均を使用してヒストグラム・テンプレートを調整することもできます。つまり、真の平均とヒストグラムから計算された平均とを比較して、ヒストグラムからの平均が真の平均から外れている場合、自分のデータにとってより適切なビン値のセットを使用して、対応するヒストグラムのヒストグラム・テンプレートを変更することを考慮できます。

注: EXTENDED 統計は、より詳細なパフォーマンスのモデル化に役立ちます。154 ページの『ワークロード管理のパフォーマンスのモデル化』を参照してください。

- ネストなしコーディネーター・アクティビティーの発生間隔時間の平均 (**coord_act_interarrival_time_avg**)。この統計は、このサービス・クラスまたはワーク・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・アクティビティーが発生してから、次のコーディネーター・アクティビティーが発生するまでの時間の算術平均を判別するために使用します。統計が最後にリセットされた時からの平均が計算されます。
- コーディネーター・アクティビティーの平均見積コスト (**coord_act_est_cost_avg**)。この統計は、このサービス・サブクラスまたはワーク・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、最後に統計がリセットされた時以降の見積もりコストの算術平均を判別するために使用します。
- アクティビティー発生間隔時間のヒストグラム。このヒストグラムは、ネストなしコーディネーター・アクティビティーの発生間隔時間を収集します。このヒストグラムを使用して、ネストなしコーディネーター・アクティビティーの発生間

隔時間分布を入手します。このデータは、システムのモデル化に、あるいはパフォーマンス・モデル化アプリケーションへの入力に役立ちます。

- アクティビティー見積コストのヒストグラム。このヒストグラムは、ネストなしコーディネーター・アクティビティーの見積コストを収集します。このヒストグラムを使用して、概算のサービス時間分布を入手します。このデータは、システムのモデル化に、あるいはパフォーマンス・モデル化アプリケーションへの入力に役立ちます。

以下の表は、ワークロード管理オブジェクトごとに収集される統計に関するクイック・リファレンスとなります。一部の統計は、常にオブジェクトごとに収集されます。他の統計は、特定の COLLECT AGGREGATE オプションが指定された場合にのみ収集されます。集約アクティビティー統計の場合、COLLECT AGGREGATE ACTIVITY DATA EXTENDED が指定されると、すべての BASE 集約アクティビティー統計も収集されます。

表 16. ワークロード管理オブジェクトごとに収集される統計

オブジェクト・タイプ	デフォルトで収集される統計	COLLECT AGGREGATE ACTIVITY DATA BASE を指定した場合に収集される統計	COLLECT AGGREGATE ACTIVITY DATA EXTENDED を指定した場合に収集される統計	COLLECT AGGREGATE REQUEST DATA BASE を指定した場合に収集される統計
しきい値キュー	• **queue_assignments_total** • **queue_size_top** • **queue_time_total**	N/A	N/A	N/A
サービス・サブクラス	• **coord_act_completed_total** • **coord_act_rejected_total** • **coord_act_aborted_total** • **concurrent_act_top** • **num_requests_active**	• **cost_estimate_top** • **rows_returned_top** • **temp_tablespace_top** • **coord_act_lifetime_top** • **request_exec_time_avg** • **coord_act_lifetime_avg** • **coord_act_exec_time_avg** • **coord_act_queue_time_avg** • アクティビティーの存続期間のヒストグラム • アクティビティー実行時間のヒストグラム • アクティビティー・キュー時間のヒストグラム • 要求実行時間のヒストグラム	• **coord_act_est_cost_avg** • **coord_act_interarrival_time_avg** • アクティビティー発生間隔時間のヒストグラム • アクティビティー見積コストのヒストグラム	• **request_exec_time_avg** • 要求実行時間のヒストグラム
サービス・スーパークラス	• **concurrent_connection_top**	N/A	N/A	
ワークロード	• **concurrent_wlo_top** • **concurrent_act_top** • **coord_act_completed_total** • **coord_act_rejected_total** • **coord_act_aborted_total** • **wlo_completed_total**	N/A	N/A	

表 16. ワークロード管理オブジェクトごとに収集される統計 (続き)

オブジェクト・タイプ	デフォルトで収集される統計	COLLECT AGGREGATE ACTIVITY DATA BASE を指定した場合に収集される統計	COLLECT AGGREGATE ACTIVITY DATA EXTENDED を指定した場合に収集される統計	COLLECT AGGREGATE REQUEST DATA BASE を指定した場合に収集される統計
ワーク・クラス (ワーク・アクションを介した)	• **act_total**	• **cost_estimate_top** • **rows_returned_top** • **temp_tablespace_top** • **coord_act_lifetime_top** • **coord_act_lifetime_avg** • **coord_act_exec_time_avg** • **coord_act_queue_time_avg** • アクティビティーの存続期間のヒストグラム • アクティビティー実行時間のヒストグラム • アクティビティー・キュー時間のヒストグラム	• **coord_act_est_cost_avg** • **coord_act_interarrival_time_avg** • アクティビティー発生間隔時間のヒストグラム • アクティビティー見積コストのヒストグラム	

ワークロード管理のヒストグラム

ヒストグラム はビンのコレクションです。ビンは、データの離散的範囲を収集するためのコンテナーです。ヒストグラムは、様々なワークロード分析およびパフォーマンスのチューニング・タスクに役立ちます。

DB2 ワークロード管理のヒストグラムは 41 個のビンを持っています。ビンの数は固定されています。40 番目のビンにはヒストグラムの最高の定義値が含まれますが、41 番目のビンは最高の定義値を超える値のためのものです。以下の図は、アクティビティー存続時間のヒストグラムを棒グラフに作図したものを示しています。

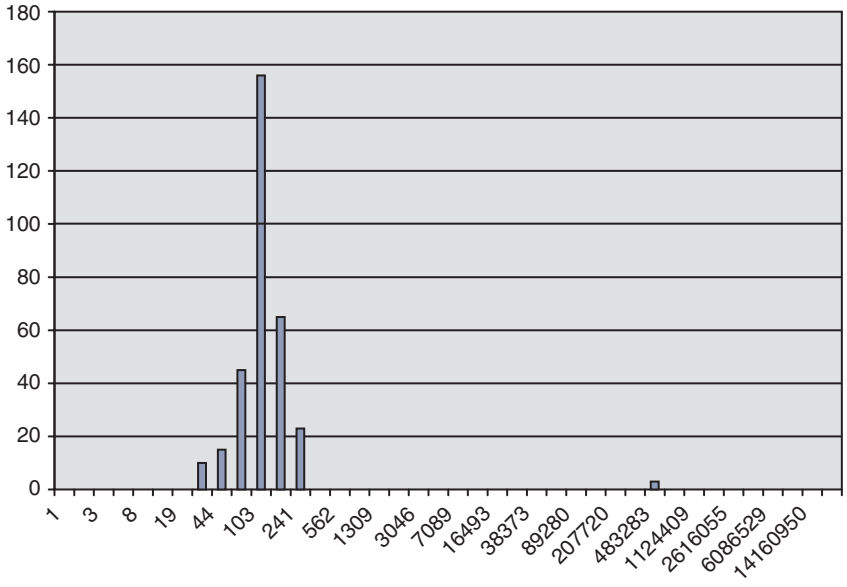


図 24. 棒グラフに作図したヒストグラム

作図されたアクティビティーの存続時間のヒストグラムは以下のデータに対応しています。それぞれのカウントは、存続時間 (ミリ秒) がビンの下限値からビンの上限値の範囲内に該当するアクティビティーの数を表しています。例えば、156 個のアクティビティーは、存続時間が 68 ミリ秒から 103 ミリ秒の範囲内でした。

```
 ビンの下限   ビンの上限   カウント
       0          1      0
       1          2      0
       2          3      0
       3          5      0
       5          8      0
       8         12      0
      12         19      0
      19         29     10
      29         44     15
      44         68     45
      68        103    156
     103        158     65
     158        241     23
     241        369      0
     369        562      0
     562        858      0
     858       1309      0
    1309       1997      0
    1997       3046      0
    3046       4647      0
    4647       7089      0
    7089      10813      0
   10813      16493      0
   16493      25157      0
   25157      38373      0
   38373      58532      0
   58532      89280      0
   89280     136181      0
  136181     207720      0
  207720     316840      0
  316840     483283      3
  483283     737162      0
  737162    1124409      0
 1124409    1715085      0
 1715085    2616055      0
 2616055    3990325      0
 3990325    6086529      0
 6086529    9283913      0
 9283913   14160950      0
14160950   21600000      0
21600000   無限大      0
```

ヒストグラムを多数のさまざまな目的に使用することができます。例えば、ヒストグラムを使用して値の分布を調べたり、範囲外にある値を識別したり、または平均および標準偏差を計算したりすることができます。ワークロードのより良い理解と特徴付けにヒストグラムを使用する方法の例については、 202 ページの『例: キャパシティー・プランニング情報が使用できない場合のワークロード管理構成の調整』および 189 ページの『例: ワークロード管理構成におけるヒストグラムからの平均および標準偏差の計算』を参照してください。

パーティション・データベース環境では、ヒストグラムはデータベース・パーティションごとに収集されます。ヒストグラムの各ビンの上限値、下限値は、すべてのデータベース・パーティションで同じ値となります。ビンを使用して、パーティションごとに情報を分析することができます。すべてのデータベース・パーティションからのヒストグラムを結合することもできます。これは、対応するビンのカウントを加算して単一のヒストグラムにまとめることによって行います。こうしてデータの全体像を把握でき、全体的なヒストグラムから全体的な平均や標準偏差を計算するなどの作業にこれを使用することができます。

ヒストグラムはサービス・サブクラスおよびワーク・クラス (ワーク・アクションを介して) に使用することができます。オブジェクトの作成または変更時に COLLECT AGGREGATE ACTIVITY DATA 節の 1 つを指定すると、これらのオブジェクトのためにヒストグラムが収集されます。ワーク・クラスについては、COLLECT AGGREGATE ACTIVITY DATA ワーク・アクションがワーク・クラスに適用されるとヒストグラムが収集されます。以下のヒストグラムが使用可能です。

- ネストなしコーディネーター・アクティビティー存続時間 (サービス・サブクラスまたはワーク・クラスに適用されたワーク・アクションに対して AGGREGATE ACTIVITY DATA BASE または AGGREGATE ACTIVITY DATA EXTENDED を指定する時)。
- ネストなしコーディネーター・アクティビティー実行時間 (サービス・サブクラスまたはワーク・クラスに適用されたワーク・アクションに対して AGGREGATE ACTIVITY DATA BASE または AGGREGATE ACTIVITY DATA EXTENDED を指定する時)。
- ネストなしコーディネーター・アクティビティー・キュー時間 (サービス・サブクラスまたはワーク・クラスに適用されたワーク・アクションに対して AGGREGATE ACTIVITY DATA BASE または AGGREGATE ACTIVITY DATA EXTENDED を指定する時)。
- 要求実行時間 (サービス・クラスに対して AGGREGATE REQUEST DATA BASE を指定する時)。このヒストグラムはワーク・クラスには適用されません。
- ネストなしアクティビティー到着間隔時間のヒストグラム (サービス・サブクラスまたはワーク・クラスに適用されたワーク・アクションに対して AGGREGATE ACTIVITY DATA EXTENDED を指定する時)。
- ネストなし DML アクティビティーの見積コスト (サービス・サブクラスまたはワーク・クラスに適用されたワーク・アクションに対して AGGREGATE ACTIVITY DATA EXTENDED を指定する時)。

アクティビティー関連のヒストグラムはすべて、完了、打ち切り、または拒否されたアクティビティーを収集します。

オプションでヒストグラム・テンプレートを指定することができます。これは、あるオブジェクトのために収集される各ヒストグラムのビンの上限値を記述したものです。ヒストグラム・テンプレートは、ヒストグラムのビンの上限値を決定するために使用されます。ヒストグラム・テンプレートは、特定のヒストグラムの外見を指定する*単位なし* のオブジェクトです。(「単位なし」とは、ヒストグラム・テンプレートに、事前に定義された計算単位が割り当てられていないことを意味します)。

サービス・サブクラスまたはワーク・アクションを作成または変更する時、適切な HISTOGRAM TEMPLATE キーワードを使用することにより、ヒストグラム・テンプレートを適用することができます。ヒストグラム・テンプレートを指定しない場合、デフォルトのテンプレート SYSDEFAULTHISTOGRAM が使用されます。AGGREGATE ACTIVITY DATA コレクションがオブジェクトに対して使用可能になっていない場合、ヒストグラム・テンプレートは無視されます。

CREATE HISTOGRAM TEMPLATE ステートメントを使用すること、および最大のビンの上限値を指定することによって、ヒストグラム・テンプレートを作成するこ

とができます。その他のビンはすべて、ビンの上限値に向かって指数関数的に増加する値として、自動的に定義されます。例えば、ビンの上限値を 3,000,000 としてヒストグラム・テンプレートを作成するには、以下のようなステートメントを発行します。

```
CREATE HISTOGRAM TEMPLATE TEMPLATE1 HIGH BIN VALUE 3000000
```

このステートメントは、以下のビンの値を使用してヒストグラム・テンプレートを作成します。

```
ビンの下限   ビンの上限
      0         1
      1         2
      2         3
      3         4
      4         6
      6         9
      9        13
     13        19
     19        28
     28        41
     41        60
     60        87
     87       127
    127       184
    184       268
    268       389
    389       565
    565       821
    821      1192
   1192      1732
   1732      2514
   2514      3651
   3651      5300
   5300      7696
   7696     11173
  11173     16222
  16222     23553
  23553     34196
  34196     49649
  49649     72084
  72084    104657
 104657    151948
 151948    220609
 220609    320297
 320297    465030
 465030    675163
 675163    980250
 980250   1423197
1423197   2066299
2066299   3000000
3000000 無限大
```

次に、既存のヒストグラムのために、このヒストグラム・テンプレートを使用することができます。例えば、サービス・スーパークラス MYSUPERCLASS 下のサービス・サブクラス MYSUBCLASS のアクティビティーの存続時間のヒストグラムのために TEMPLATE1 ヒストグラム・テンプレートを使用するには、以下のようなステートメントを発行します。

```
ALTER SERVICE CLASS MYSUBCLASS UNDER MYSUPERCLASS
ACTIVITY LIFETIME HISTOGRAM TEMPLATE TEMPLATE1
```

ALTER SERVICE CLASS ステートメントをコミットした後、このサービス・サブクラスに関して収集されるアクティビティー存続時間のヒストグラムは、SYSDEFAULTHISTOGRAM ヒストグラム・テンプレートからのデフォルトのビンの上限値を使用する代わりに、TEMPLATE1 ヒストグラム・テンプレートによって決定されたビンの上限値を使用します。

注: 別のヒストグラム・テンプレートを使用するようサービス・クラスを変更する場合、あるいはヒストグラム・テンプレートを変更する場合、その変更は統計のリセットが発生するまで有効になりません。

DROP HISTOGRAM TEMPLATE ステートメントを使用してヒストグラム・テンプレートをドロップすることができます。

SYSCAT.HISTOGRAMTEMPLATES ビューを照会することによってヒストグラム・テンプレートを表示できます。SYSCAT.HISTOGRAMTEMPLATEBINS ビューを照会することによって対応するヒストグラム・テンプレートのビンの上限値を表示することができます (ビンの下限値は、常に前のビンの上限値か 0 (最初のビンの場合) になります)。

統計イベント・モニターを使用したワークロード管理統計の収集

ワークロード管理オブジェクトの統計は、統計イベント・モニターに送信して、履歴分析できます。

統計を使用して、長期にわたるシステムの動作を理解し (例えばアクティビティーの平均存続時間、アクティビティーがキューで費やされた時間、小さなアクティビティーと比較した大きなアクティビティーの分散など)、しきい値を設定し (例えば並行アクティビティーの上限を見つけるなど)、問題を検出する (例えば現在ユーザーが経験している平均存続時間が通常より長いのかどうかを検出するなど) ことができます。各ワークロード管理オブジェクトについてどの統計が収集されるのかについての説明は、134 ページの『ワークロード管理オブジェクトの統計』を参照してください。

ワークロード管理統計は、決められた時間間隔でイベント・モニターへ自動送信することも、任意の時点で手動で送信することもできます。

ワークロード管理統計を決められた時間間隔で自動収集するには、次のようにします。

1. CREATE EVENT MONITOR ステートメントを使用して、STATISTICS イベント・モニターを作成します。 例えば、次のステートメントを発行できます。

   ```
   CREATE EVENT MONITOR STATS1 FOR STATISTICS WRITE TO TABLE
   ```

2. COMMIT ステートメントを使用して、変更をコミットします。
3. SET EVENT MONITOR STATE ステートメントを使用して、イベント・モニターを活動化します。 SET EVENT MONITOR STATE ステートメントを使用せずに、STATISTICS イベント・モニターに AUTOSTART のデフォルトを使用して、データベースが次回活動化されたときにこのイベント・モニターが活動化されるようにする方法もあります。ただし、データベース・パーティションで一度にアクティブにできる STATISTICS タイプのイベント・モニターは 1 つだけです。複数の STATISTICS イベント・モニターを定義する場合は、AUTOSTART オプションを使用しないでください。

4. COMMIT ステートメントを使用して、変更をコミットします。
5. オプション: その他の統計の収集を使用可能にします。 デフォルトでは、各ワークロード管理オブジェクトについて、最小セットの統計のみが収集されます。各オブジェクトに対して、デフォルトでどの統計が収集されるかについては、 134 ページの『ワークロード管理オブジェクトの統計』を参照してください。
   ALTER SERVICE CLASS ステートメントおよび ALTER WORK ACTION SET ステートメントの COLLECT AGGREGATE ACTIVITY DATA キーワードを使用して、サービス・サブクラスとワーク・クラスの集約アクティビティー・データの収集を指定します。ALTER SERVICE CLASS ステートメントの COLLECT AGGREGATE REQUEST DATA キーワードを使用して、サービス・サブクラスの集約要求データの収集を指定します。変更があればコミットします。
6. データベース構成パラメーター **wlm_collect_int** を更新して、収集間隔を指定します。 **wlm_collect_int** パラメーターは、時間間隔を分単位で指定します。間隔ごとに、すべてのワークロード管理オブジェクトのワークロード管理統計のメモリー内コピーがアクティブな統計イベント・モニターに書き込まれ、メモリー内統計がリセットされます。パーティション・データベース環境では、**wlm_collect_int** パラメーターが、カタログ・パーティションで更新される必要があります。このパラメーターは、動的に更新できます。以下に例を示します。

```
CONNECT TO database alias
UPDATE DATABASE CONFIGURATION USING WLM_COLLECT_INT 5 IMMEDIATE
```

上記の各ステップを実行すると、ワークロード管理統計は **wlm_collect_int** 分ごとに統計イベント・モニターに書き込まれます。この統計イベント・モニターに書き込まれた各レコードは、STATISTICS_TIMESTAMP 値と LAST_WLM_RESET 値を持っています。LAST_WLM_RESET から STATISTICS_TIMESTAMP までの時間間隔により、収集の間隔 (つまり、そのレコードの統計が収集された時間間隔) が定義されます。

wlm_collect_int パラメーターがゼロ以外の値に設定され、アクティブな統計イベント・モニターがない場合、メモリー内ワークロード管理統計は引き続き **wlm_collect_int** 分ごとにリセットされますが、統計は収集されません。するとデータは失われます。このため、統計イベント・モニターを活動化せずに **wlm_collect_int** 値にゼロ以外の値を指定することはお勧めできません。

wlm_collect_int パラメーターが 0 に設定されている場合 (デフォルト)、統計が自動的に統計イベント・モニターに送信されることはありません。
WLM_COLLECT_STATS ストアード・プロシージャーを使用すると、後で履歴分析をするために、統計を統計イベント・モニターに手動で送信できます。このプロシージャーが呼び出されると、自動統計収集間隔で行われる場合と同じアクションが実行されます。つまり、メモリー内統計が統計イベント・モニターに送信されて、メモリー内統計がリセットされます。アクティブな統計イベント・モニターがない場合、メモリー内値はリセットされますが、データは収集されません。統計をリセットしようとするだけの場合は、アクティブな統計イベント・モニターがないときに WLM_COLLECT_STATS プロシージャーを呼び出すことができます。

統計の手動収集は、統計の自動収集の支障にはなりません。例えば、**wlm_collect_int** が 60 に設定されているとします。統計は、統計イベント・モニターに 1 時間ごとに送信されます。ここで、最後に統計が収集された時刻が 5:30 AM であると仮定します。WLM_COLLECT_STATS プロシージャーを 5:55 AM に呼び出すと、統計

のメモリー内値がイベント・モニターに送信され、統計がリセットされます。次の自動統計収集は、前回の自動収集の 1 時間後である 6:30 AM に行われます。収集の間隔は手動による収集およびその間隔の間に発生した統計のリセットの影響を受けません。

注: ワークロード管理統計の表関数は、メモリー内統計の現行値を報告します。自動ワークロード管理統計収集を使用可能にしている場合、これらの値は **wlm_collect_int** データベース構成パラメーターで定義された間隔で定期的にリセットされます。表関数で報告される統計を調べるときは、必ず LAST_RESET 列を検討するようにします。この列は、前回メモリー内統計がリセットされた時刻を示しています。前回のリセット時刻から現在時刻までの時間間隔が不十分な場合、意味のある結論を引き出すのに十分なデータがない場合があります。

注: ワークロード管理統計の自動収集を使用している場合は、イベント・モニターのファイルや表を定期的に整理する必要があります。イベント・モニターが収集されたデータを自動的に整理することはなく、自動収集によってファイルと表は時間とともにいっぱいになってしまいます。

注: データベースが非活動化されると、メモリー内統計はリセットされます。データベースを非活動化すると、統計は統計イベント・モニターには送信されません。前回の収集以降累積された統計を非活動化のために失いたくない場合は、データベースを非活動化する前に WLM_COLLECT_STATS プロシージャーを手動で呼び出す必要があります。

注: WLM_COLLECT_STATS プロシージャーは、RESET MONITOR コマンドとは別の方法で統計をリセットします。 RESET MONITOR コマンドは、現在の値を保管することによってスナップショット・モニター・エレメントの値をリセットします。 RESET MONITOR コマンドが発行された後で、スナップショット処理はこれらの値と現行値との間の差分を報告します。これに対して、WLM_COLLECT_STATSプロシージャーによるリセットでは値は保管されず、該当する各ワークロード管理オブジェクトの統計カウンター自体がすべてリセットされます。

また、RESET MONITOR コマンドでは、各プロセス (アタッチ) には、モニター・データについての独自のプライベート・ビューがあります。あるユーザーがリセットを実行しても、他のユーザーは影響を受けません。これに対して、ワークロード・マネージャー統計のリセットは、すべてのユーザーに適用されます。

統計を取得するためのワークロード管理の表関数

このトピックで説明されている表関数を使用して、ワークロード管理オブジェクトについての統計を取得することができます。

ワークロード管理の表関数は SYSPROC スキーマに用意されています。これらの表関数はハイパフォーマンスであり、現在実行中のワークロードにほとんど影響を与えることなく、システムで実行されている作業に関する情報を返すことができます。

統計は、例えばワークロード管理の構成への変更が、期待された効果をもたらしたかどうかを検証するなど、さまざまな異なる目的に使用できます。例えば、READ アクティビティーを分類するための新規のワーク・クラスを作成した場合、READ

アクティビティーが新規のワーク・クラスの下に分類されているかどうかを検証したいと思うかもしれません。また、表関数を使用してシステムに関する特定の問題をすぐに認識することができます。例えば、表関数を使用して平均的なアクティビティーの存続時間の許容値を判別することや、この値が通常の範囲を超えていて、さらに調査の必要な問題を指し示している可能性がある場合に、それを認識することができます。

すべての統計表関数は、統計が最後にリセットされた時から累算した統計を返します。

dbpartitionnum 変数については、-2 を指定してすべてのデータベース・パーティションからデータを収集することを指示することもできますし、あるいは -1 を指定して、表関数呼び出しを発行したアプリケーションが接続されたデータベース・パーティション (つまり、コーディネーター・パーティション) からのみデータを収集することを指示することもできます。これらの表関数をアプリケーション・プログラムから呼び出す場合、sqlmon.h ヘッダー・ファイル内の SQLM_CURRENT_NODE および SQLM_ALL_NODE 定数を使用すべきであり、-1 および -2 をリテラルとして使用することを避けるべきです。

- WLM_GET_SERVICE_SUPERCLASS_STATS(*service_superclass_name*, *dbpartitionnum*)。この表関数を使用して、統計が最後にリセットされた時から計算された、同時接続の高水準点に関する情報を取得することができます。ワイルドカード文字を使用すると、複数のサービス・スーパークラスおよびデータベース・パーティションをまたぐことができます。
- WLM_GET_SERVICE_SUBCLASS_STATS(*service_superclass_name*, *service_subclass_name*, *dbpartitionnum*)。この表関数を使用して、統計が最後にリセットされてから計算されたアクティビティーの数および平均実行時間などの要約された統計を取得することができます。1 つ以上のデータベース・パーティションにまたがる複数のサービス・サブクラスのためにこの情報を取得できます。ワイルドカード文字を使用すると、複数のサービス・スーパークラス、サービス・サブクラス、およびデータベース・パーティションをまたぐことができます。この表関数の使用についての詳しい情報は、186 ページの『例: サービス・クラスからの特定時点の統計の取得』、 187 ページの『例: ワークロード管理の表関数を使用したデータの集約』、 190 ページの『例: サービス・クラス関連のシステム・スローダウンの分析』、および 192 ページの『例: ワークロード関連のシステム・スローダウンの調査』を参照してください。
- WLM_GET_WORKLOAD_STATS(*workload_name*, *dbpartitionnum*)。この表関数を使用して、1 つまたはすべてのワークロードおよびデータベース・パーティションの要約された統計を取得することができます。ワイルドカード文字を使用すると、複数のワークロード名およびデータベース・パーティションをまたぐことができます。返される統計には、統計のリセット以後に計算された、完了したアクティビティーに関する情報、および高水準点に関する情報が含まれます。この表関数の使用についての詳しい情報は、192 ページの『例: ワークロード関連のシステム・スローダウンの調査』を参照してください。
- WLM_GET_WORK_ACTION_SET_STATS(*work_action_set_name*, *dbpartitionnum*)。この表関数を使用して、1 つ以上のデータベース・パーティションにまたがる、1 つ以上のワーク・アクション・セットに関する要約された統計を取得することができます。返される統計には、統計が最後にリセットされてから、それぞれのワーク・クラスに割り当てられたアクティビティーの数についての情報が含まれま

す。この表関数の使用については、 193 ページの『例: アクティビティー・タイプごとのワークロードの分析』を参照してください。

- WLM_GET_QUEUE_STATS (*threshold_predicate*, *threshold_domain*, *threshold_name*, *threshold_id*)。この表関数を使用して、しきい値キューに関する情報を取得することができます。この情報から、現在何個のアクティビティーがキューに入れられているか、またアクティビティーが最後にキューから出た時刻を知ることができます。統計情報には、キューでアクティビティーが費やした平均時間やキューのサイズの高水準点などの情報も含まれます。この情報を、ある特定のキューがシステム内でボトルネックになっていないかどうか (つまり、アクティビティーがこのキューで費やす時間が長すぎないか) を判別するために使用することができます。

ワークロード管理オブジェクトの統計のリセット

このトピックは、ワークロード管理オブジェクトの統計をリセットする方法を説明します。

4 つのイベントによって、それぞれのワークロード管理オブジェクトのために保管されているメモリー内の統計がリセットされます。(それぞれのオブジェクトのために維持されている統計についての説明は、 134 ページの『ワークロード管理オブジェクトの統計』を参照してください。)

- WLM_COLLECT_STATS ストアード・プロシージャーが呼び出される。詳しくは、 144 ページの『統計イベント・モニターを使用したワークロード管理統計の収集』を参照してください。
- **wlm_collect_int** データベース構成パラメーターにより制御される、自動のワークロード管理統計収集およびリセットのプロセスによって、収集およびリセットが起こる。詳しくは、 144 ページの『統計イベント・モニターを使用したワークロード管理統計の収集』を参照してください。
- データベースが再活動化される。データベース・パーティション上でデータベースが活動化されるたびに、そのデータベース・パーティション上のすべてのワークロード管理オブジェクトの統計はリセットされます。
- 統計が維持されているオブジェクトが変更され、その変更内容がコミットされる。例えば、あるサービス・サブクラスが変更される場合、ALTER がコミットされるとそのサービス・サブクラスのメモリー内の統計はリセットされます。

統計の表関数を使用したり、LAST_RESET 列にあるタイム・スタンプを見たりして、特定のワークロード管理オブジェクトの統計が最後にリセットされた時間を判別することができます。例えば、SYSDEFAULTUSERCLASS サービス・スーパークラス下のサービス・サブクラス SYSDEFAULTSUBCLASS に対して最後に統計がリセットされた時間を知るために、以下のような照会を発行できるかもしれません。

```
SELECT LAST_RESET FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS( 'SYSDEFAULTUSERCLASS',
'SYSDEFAULTSUBCLASS', -2)) AS T
```

すべての統計表関数は、統計が最後にリセットされた時から累算した統計を返します。統計のリセットは、データベースがアクティブにされた時または非アクティブにされた時、ワークロード管理オブジェクトを変更した時 (そのオブジェクトの統計だけがリセットされます)、そして WLM_COLLECT_STATS ストアード・プロシージャーを呼び出した時に発生します。**wlm_collect_int** データベース構成パラメー

ターをゼロ以外の値に設定した場合にも、統計はこのパラメーターによって定義される期間に従って自動的にリセットされます。

wlm_collect_int によって指定される期間は、この構成パラメーターによって指定されるインターバルの間に発生する統計のリセットによる影響を受けません。例えば、**wlm_collect_int** によって指定される 20 分間のインターバルが開始してから 5 分後に WLM_COLLECT_STATS 表関数を実行する場合、そのインターバルはやはり 15 分後に有効期限が切れます。統計の収集およびリセットが発生することによって、次の統計の収集とリセットの発生が 5 分遅くなることはありません。

別のヒストグラム・テンプレートを使用するようサービス・クラスを変更する場合、あるいはヒストグラム・テンプレートを変更する場合、その変更は統計のリセットが発生するまで有効になりません。

WLM_COLLECT_STATS 表関数を呼び出して統計の収集とリセットを行うとき、これと同時に別の収集とリセットが進行中であると (例えば、**wlm_collect_int** による定期的な収集およびリセットと表関数の呼び出しとがオーバーラップする場合、あるいは別のユーザーが同時に WLM_COLLECT_STATS を呼び出す場合)、WLM_COLLECT_STATS からの収集およびリセット要求は無視され、警告 SQL1632W が返されます。

しきい値違反のモニター

ワークロード管理のしきい値の違反があった場合には、アクティブな THRESHOLD VIOLATIONS イベント・モニターがある場合、それにしきい値違反レコードが書き込まれます。

しきい値違反レコードには、以下の情報が含まれます。

- 違反のあったしきい値の説明 (ID、最大値、など)。
- しきい値に違反したアクティビティーの ID。これには、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の ID、アクティビティーの固有 ID、および作業単位 ID が含まれます。
- しきい値の違反が生じた時刻。
- 取られた処置。この処置は、しきい値に違反したアクティビティーが続行を許可されたか、または停止されたかを示します。アクティビティーが停止された場合、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 エラーを受信したはずです。

オプションで、アクティビティーによってしきい値違反が起こった場合に、詳細なアクティビティー情報 (ステートメント・テキストを含む) がアクティブなアクティビティー・イベント・モニターに書き込まれるようにすることができます。アクティビティー情報は、しきい値の違反が生じたときではなく、アクティビティーの完了時に書き込まれます。しきい値の違反時にアクティビティー情報を収集するように指定するには、COLLECT ACTIVITY DATA キーワードを CREATE または ALTER しきい値またはワーク・アクション・セットのステートメントで使用します。

しきい値の違反をモニターするには、以下のようにします。

1. CREATE EVENT MONITOR ステートメントを使用して、タイプが THRESHOLD VIOLATIONS であるイベント・モニターを作成します。 以下に例を示します。

   ```
   CREATE EVENT MONITOR VIOLATIONS FOR THRESHOLD VIOLATIONS WRITE TO TABLE
   ```

2. COMMIT ステートメントを使用して、変更をコミットします。
3. SET EVENT MONITOR STATE ステートメントを使用して、イベント・モニターを活動化します。 SET EVENT MONITOR STATE ステートメントを使用せずに、THRESHOLD VIOLATIONS イベント・モニターに AUTOSTART のデフォルトを使用して、データベースが次回活動化されたときにこのイベント・モニターが活動化されるようにする方法もあります。ただし、データベース・パーティションで一度にアクティブにできる THRESHOLD VIOLATIONS タイプのイベント・モニターは 1 つだけです。複数の THRESHOLD VIOLATIONS イベント・モニターを定義する場合は、AUTOSTART オプションを使用しないでください。
4. COMMIT ステートメントを使用して、変更をコミットします。

注: しきい値を作成する場合には、しきい値違反が生じたときにモニターできるよう、しきい値違反イベント・モニターを作成してアクティブにすることをお勧めします。しきい値違反イベント・モニターが与える影響は、しきい値違反が生じない限り、全くありません。

個々のアクティビティーのデータ収集

ACTIVITIES イベント・モニターを使用して、システム上で実行される個々のアクティビティーのデータを収集することができます。収集されるデータには、ステートメント・テキストおよびコンパイル環境といった項目が含まれており、問題の調査と診断に使用したり、他のツール (例えば、設計アドバイザー) への入力データとして使用したりすることができます。

サービス・サブクラス、ワークロード、ワーク・クラス (ワーク・アクションを通して)、およびしきい値違反に対して、個々のアクティビティーに関する情報を収集できます。これらのワークロード管理オブジェクトに対するアクティビティーの収集は、CREATE または ALTER ステートメントの COLLECT ACTIVITY DATA キーワードを使用して有効にします。以下の場合に、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は、アクティブな ACTIVITIES イベント・モニターに送信されます。

- アクティビティーは、COLLECT ACTIVITY DATA が指定されているワークロードにマップされたアプリケーションによってサブミットされた。
  - または、アクティビティーは、COLLECT ACTIVITY DATA が指定されているサービス・サブクラス内で実行されている。
  - または、アクティビティーには COLLECT ACTIVITY DATA ワーク・アクションが適用されている。
  - または、アクティビティーは、COLLECT ACTIVITY DATA アクションで定義されたしきい値に違反する。

COLLECT ACTIVITY DATA キーワードは、ACTIVITIES イベント・モニターへ送信される情報量もコントロールします。このキーワードで WITH DETAILS が指定

されている場合、ステートメント情報 (ステートメント・テキストなど) が収集されます。キーワードで WITH DETAILS AND VALUES が指定されている場合、データ値も収集されます。

アクティビティーには、複数の COLLECT ACTIVITY DATA キーワードが適用される場合があります。例えば、アクティビティーは、COLLECT ACTIVITY DATA が指定されているサービス・サブクラス内で実行され、実行中に COLLECT ACTIVITY DATA アクションを持つしきい値に違反するかもしれません。この場合は、アクティビティーは 1 度のみ収集されます。アクティビティーには、最も大量の情報の収集を指定する COLLECT キーワードが適用されます。例えば、COLLECT ACTIVITY DATA WITHOUT DETAILS と COLLECT ACTIVITY DATA WITH DETAILS の両方が 1 つのアクティビティーに適用された場合、アクティビティーは詳細情報と共に収集されます。

特定のワークロード管理オブジェクトに対するアクティビティーの収集を有効にするには、以下のようにします。

1. CREATE EVENT MONITOR ステートメントを使用して、ACTIVITIES イベント・モニターを作成します。
2. COMMIT ステートメントを使用して、変更をコミットします。
3. SET EVENT MONITOR STATE ステートメントを使用して、イベント・モニターを活動化します。 SET EVENT MONITOR STATE ステートメントを使用せずに、ACTIVITIES イベント・モニターに AUTOSTART のデフォルトを使用して、データベースが次回活動化されたときにこのイベント・モニターが活動化されるようにする方法もあります。ただし、データベース・パーティションで一度にアクティブにできる ACTIVITIES タイプのイベント・モニターは 1 つだけです。複数の ACTIVITIES イベント・モニターを定義する場合は、AUTOSTART オプションを使用しないでください。
4. COMMIT ステートメントを使用して、変更をコミットします。
5. ALTER SERVICE CLASS、ALTER WORK ACTION SET、ALTER THRESHOLD、または ALTER WORKLOAD ステートメントを使用してどのオブジェクトに対してアクティビティーを収集するかを示し、COLLECT ACTIVITY DATA キーワードを指定します。
6. COMMIT ステートメントを使用して、変更をコミットします。

注: 個々のアクティビティー収集は、ワークロード管理統計収集より高コストです。可能な限り少数のアクティビティーを収集するようアクティビティー収集をセットアップしてください。例えば、特定のアプリケーションによってサブミットされたアクティビティーを調査する必要がある場合、そのアプリケーション専用のワークロードまたはサービス・クラスを作成してそのアプリケーションを隔離し、そのワークロードまたはサービス・クラスに対する収集のみを有効にすることができます。

アクティビティーをキャプチャーする必要が常に事前にわかるとは限りません。例えば、ある照会の実行に長時間かかっている場合、後に分析するためにその情報を収集する必要があるかもしれません。この場合、アクティビティーが既にシステムに入っているため、このワークロード管理オブジェクトに対して COLLECT ACTIVITY DATA キーワードを指定するには遅すぎます。この場合には、WLM_CAPTURE_ACTIVITY_IN_PROGRESS ストアード・プロシージャーを使用で

きます。 WLM_CAPTURE_ACTIVITY_IN_PROGRESS ストアード・プロシージャーは、実行しているアクティビティーに関する情報をアクティブな ACTIVITIES イベント・モニターに送信します。アプリケーション・ハンドル、作業単位 ID、およびアクティビティー ID を使用して、収集するアクティビティーを指定します。プロシージャーが呼び出されると、このアクティビティーに関する情報は直ちに ACTIVITIES イベント・モニターに送信されます。アクティビティーの完了まで待つ必要はありません。

WLM_CAPTURE_ACTIVITY_IN_PROGRESS ストアード・プロシージャーを使用して INOUT パラメーターを持つプロシージャーのアクティビティー情報を収集する場合、キャプチャーが生じる前に INOUT の値が上書きされることがあります。サービス・サブクラス、ワークロード、ワーク・アクション、または予測しきい値に対する COLLECT ACTIVITY DATA WITH DETAILS AND VALUES キーワードの結果としてアクティビティーが収集される場合、この状態は生じません。

設計アドバイザーへのアクティビティー情報のインポート

アクティビティー・イベント・モニターによって収集されたアクティビティーを設計アドバイザーにインポートして、これらアクティビティーがアクセスするデータベース・オブジェクトについての決定をする際に役立てることができます。

設計アドバイザーにインポートされるアクティビティーは、必ず、COLLECT ACTIVITY DATA WITH DETAILS オプションまたは COLLECT ACTIVITY DATA WITH DETAILS AND VALUES オプションを使用して収集されたものです。COLLECT ACTIVITY DATA WITHOUT DETAILS オプションは不十分で、設計アドバイザーが必要とするステートメント・テキストをキャプチャーしません。

アクティビティー情報をアクティビティー・イベント・モニター表から設計アドバイザーにインポートするには、db2advis コマンドに **-wlm** パラメーターを付け、その後に以下のようなその他のパラメーターを付けて実行します。

1. アクティビティー・イベント・モニター名
2. オプション: ワークロード名またはサービス・クラス名
3. オプション: 開始時刻と終了時刻

例えば、SAMPLE データベースで DB2ACTIVITIES イベント・モニターによって収集されたすべてのアクティビティーに関する情報をインポートするには、次のコマンドを使用します。

```
db2advis -d SAMPLE -wlm DB2ACTIVITIES
```

注: 設計アドバイザーのコマンド行インターフェースでは、アクティビティー・イベント・モニター表からの情報しかインポートできません。

アクティビティーのキャンセル

アクティビティーが消費するリソースが多すぎる場合、またはハングした場合に、そのアクティビティーをキャンセルする場合があります。アクティビティーのキャンセルの方が、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を強制終了するよりも緩やかな対応です。キャンセルされたアクティビティーにより、ユーザーに SQL4725N が戻されますが、接続を終了したり、そのユーザーの他のアクテ

ィビティーに影響を与えたりはしません。アプリケーションを強制終了すると、そのユーザーの接続とアクティビティーの両方が終了します。

アクティビティーを明示的にキャンセルできるのは、コーディネーター・アクティビティーが現在そのアクティビティーの要求を処理している場合のみです。IDLE 状態 (つまり、処理されている要求がない状態) でアクティビティーをキャンセルした場合、アクティビティーは CANCEL_PENDING 状態に置かれ、次に受信される要求でキャンセルされます。例えば、CURSOR アクティビティーをフェッチの間にキャンセルしようとすると、SQL4725N エラーはキャンセルの後の次のフェッチまでユーザーに戻されません。

ロード・ユーティリティーおよびストアード・プロシージャーを含むすべてのユーザー・アクティビティーはキャンセル可能です。

アクティビティーをキャンセルするには、以下のようにします。

1. キャンセルするアクティビティーを識別します。
   WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数を使用して、アプリケーションで実行されているアクティビティーを識別できます。
   WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 内の情報が、アクティビティーが実行している作業を識別するのに十分でない場合は、
   WLM_GET_ACTIVITY_DETAILS 表関数を使用して特定のアクティビティーに関する詳細をさらに表示することもできます。
2. WLM_CANCEL_ACTIVITY ストアード・プロシージャーを使用してアクティビティーをキャンセルします。 ストアード・プロシージャーには、以下の引数をとります。*application_handle*、*uow_id*、および *activity_id*。このストアード・プロシージャーの使用方法は、194 ページの『例: ハング・アクティビティーの識別』の例を参照してください。

不良アクティビティーに関する情報のキャプチャーと調査についてのガイドライン

このトピックでは、不良アクティビティーに関する情報のキャプチャーと調査についてのガイドラインが提供されています。

最初に、不良アクティビティーであると見なす一連の基準を設定します。以下に例を示します。

- 見積コストの小さいアクティビティー用のサービス・クラスで実行されているものの、1 時間を超えて実行されているアクティビティー
- 異常なほど大量の行を戻すアクティビティー
- TEMPORARY 表スペースを異常に大量に消費するアクティビティー

次に、こうした基準を記述したしきい値を作成して、COLLECT ACTIVITY DATA WITH DETAILS アクションを組み込みます。そのしきい値に違反すると、アクティビティー完了時に、しきい値に違反したそのアクティビティーに関する情報がアクティブな ACTIVITIES イベント・モニターに送信されます。

例えば、3 時間を超えて実行されているデータベース・アクティビティーに関する情報を収集するには、以下のようなしきい値を作成します。

```
CREATE THRESHOLD LONGRUNNINGACTIVITIES FOR DATABASE ACTIVITIES ENFORCEMENT DATABASE
WHEN ACTIVITYTOTALTIME > 3 HOURS COLLECT ACTIVITY DATA WITH DETAILS CONTINUE
```

その後、イベント・モニターに書き込まれた情報を分析できます。また DML アクティビティーにはイベント・モニターに書き込まれたステートメント・テキストおよびコンパイル環境情報があるので、こうした情報に対して DB2 Explain を実行してアクティビティーのパフォーマンスをさらに調査することができます。

ワークロード管理のパフォーマンスのモデル化

システムのワークロードはモデル化が可能です。これは、アクティビティーの到着比率分布 (しばしば、その反対である*到着間隔* 時間分布の形で測定される) が定める比率でシステムに到着するアクティビティーのセット、およびサービス時間分布に従ってアクティビティーがシステムで実行に費やす時間の量、という形でモデル化できます。

到着間隔時間とは、1 つのアクティビティーの到着から次のアクティビティーの到着までの間の時間です。サービス時間は、アクティビティーがシステム上で実行に費やす時間です。例えば、0 秒の時点で照会をサブミットし、これがキューで 2 秒間費やし、5 秒の時点で完了する場合、サービス時間は 5 - 2 = 3 秒となります。サービス時間は、他の作業がシステム上で実行されていないことを前提としています (つまり、これは実測上の実行時間ではなく、独立してアクティビティーを実行するのに要するであろうと考えられる時間です)。 DML アクティビティーの場合、サービス時間分布は、アクティビティーの CPU 時間と入出力時間の両方を考慮に入れた見積コスト (timeron 単位) を使用して概算することができます。

システムのワークロード・モデルを作成するには、システム上のアクティビティーの到着間隔時間分布およびサービス時間分布を測定します。到着間隔時間分布およびおおよそのサービス時間分布 (見積コストを使用) は、サービス・サブクラスまたはワーク・クラス (ワーク・アクションを使用) の拡張集約アクティビティー統計、および統計イベント・モニターを使用して入手することができます。これらの統計は、デフォルトでは収集されません。詳しくは、 134 ページの『ワークロード管理オブジェクトの統計』を参照してください。

ヒストグラムの作業

ヒストグラム・テンプレートの作成

CREATE HISTOGRAM TEMPLATE ステートメントを使用してヒストグラム・テンプレートを作成します。ヒストグラム・テンプレートは、サービス・サブクラスおよびワーク・アクションにより、ヒストグラムで維持される統計の bin 値を定義するために使用されます。

ヒストグラム・テンプレートを作成するためには、SYSADM または DBADM 権限が必要です。

前提条件について詳しくは、以下のトピックを参照してください。

- 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』
- 命名規則

一部の DB2 サービス・サブクラス、ワーク・クラス・アクティビティー、および要求統計は、ヒストグラムを使用して収集されます。すべてのヒストグラムには特定数の bin があり、それぞれの bin はアクティビティーまたは要求がカウントされる範囲を表します。bin に使用される単位のタイプは、作成するヒストグラムのタイプによって異なります。ヒストグラム・テンプレートは、ヒストグラム中の最後から 2 番目の bin の最高値を表し、これはヒストグラム中のすべての bin の値に影響します。ヒストグラムについて詳しくは、 140 ページの『ワークロード管理のヒストグラム』を参照してください。

ヒストグラム・テンプレートを作成するには、次のようにします。

1. 作成するヒストグラム・テンプレートの名前と、最後から 2 番目の bin の最高値を設定する HIGH BIN VALUE キーワードの値を指定して、CREATE HISTOGRAM TEMPLATE ステートメントを発行します。
2. 変更をコミットします。 変更をコミットすると、ヒストグラムは SYSCAT.HISTOGRAMTEMPLATES ビューに追加され、bin は SYSCAT.HISTOGRAMTEMPLATEBINS ビューに追加されます。

ヒストグラム・テンプレートの変更

既存のヒストグラム・テンプレートを変更するには、ALTER HISTOGRAM TEMPLATE ステートメントを使用します。ヒストグラム・テンプレートは、サービス・サブクラスおよびワーク・アクションにより、ヒストグラムで維持される統計のビン値を定義するために使用されます。

ヒストグラム・テンプレートを変更するためには、SYSADM または DBADM 権限が必要です。

前提条件について詳しくは、 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』を参照してください。

一部の DB2 サービス・サブクラス、ワーク・クラス・アクティビティー、および要求統計は、ヒストグラムを使用して収集されます。すべてのヒストグラムには特定数のビンがあり、それぞれのビンはアクティビティーまたは要求がカウントされる範囲を表します。ビンに使用される単位のタイプは、作成するヒストグラムのタイプによって異なります。ヒストグラム・テンプレートは、ヒストグラム中の最後から 2 番目のビンの最高値を表し、これはヒストグラム中のすべてのビンの値に影響します。ヒストグラムについて詳しくは、 140 ページの『ワークロード管理のヒストグラム』を参照してください。

ヒストグラム・テンプレートを変更するには、次のようにします。

1. ALTER HISTOGRAM TEMPLATE ステートメントを発行し、変更するヒストグラム・テンプレートの名前と、最後から 2 番目のビンの最高値を変更するための HIGH BIN VALUE パラメーターの値を指定します。
2. 変更をコミットします。 変更をコミットすると、ヒストグラムのビンの最高値が SYSCAT.HISTOGRAMTEMPLATEBINS ビューで更新されます。この変更は、ワークロード管理統計が次回リセットされるまで有効になりません。詳しくは、 148 ページの『ワークロード管理オブジェクトの統計のリセット』を参照してください。

3. オプション: WLM_COLLECT_STATS ストアード・プロシージャーを実行して統計を収集およびリセットし、新しいヒストグラム・テンプレートがただちに使用されるようにします。

ヒストグラム・テンプレートのドロップ

ヒストグラム・テンプレートは、必要ではなくなった場合にドロップできます。

ヒストグラム・テンプレートをドロップするためには、SYSADM または DBADM 権限が必要です。

前提条件について詳しくは、 313 ページの『付録 A. ワークロード管理 DDL ステートメントの考慮事項』を参照してください。

SYSDEFAULTHISTOGRAM ヒストグラム・テンプレートは、ドロップすることはできません。

サービス・サブクラスまたはワーク・アクションに参照されるヒストグラム・テンプレートはドロップできません。ヒストグラム・テンプレートを参照するサービス・サブクラスおよびワーク・アクションは、SYSCAT.HISTOGRAMTEMPLATESUSE ビューを照会して表示できます。

ヒストグラム・テンプレートをドロップするには、次のようにします。

1. DROP HISTOGRAM TEMPLATE ステートメントを使用します。
2. 変更をコミットします。 変更をコミットすると、ヒストグラムは SYSCAT.HISTOGRAMTEMPLATES ビューから除去され、その bin は SYSCAT.HISTOGRAMTEMPLATEBINS ビューから除去されます。

第 4 部 例

© Copyright IBM Corp. 2007, 2008

第 7 章 ワークロード管理の例

例: サービス・クラスの使用

以下の例では、サービス・クラスを使用してデータベース・ワークロードを制御する方法について示します。

この例は、International Beer Emporium という架空の企業を使用して説明されます。International Beer Emporium は販売、会計、技術、検査、および生産という 5 つの主要な部門で構成されている中堅企業です。 5 つの部門はすべて、同じ製品カタログ・データベースを共有しています。

ワークロード管理ソリューションの初期インプリメンテーション

製品カタログ・データベースは、ほとんどいつも良好に稼働しています。ただし、ユーザーから、最大接続数を超えたためにアプリケーションがデータベースに接続できないという苦情が上がることがあります。 DB2 バージョン 9.5 へのマイグレーション後に、データベース管理者の Bob がサービス・クラスを試してみることにします。 Bob は、5 つの部門ごとに製品カタログ・データベースの使用パターンを調べ、データベースの接続が時々不足する理由を確かめたいと思っています。以下に、Bob がサービス・クラスをセットアップする場合に従うステップを示します。

1. まず最初に、Bob は各部門のサービス・スーパークラスを作成します (各サービス・スーパークラスごとにデフォルトのサービス・サブクラスも自動的に作成されます)。
   - 販売部門には SALES が作成されます。
     ```
     CREATE SERVICE CLASS SALES
     ```
   - 会計部門には ACCOUNTING が作成されます。
     ```
     CREATE SERVICE CLASS ACCOUNTING
     ```
   - 技術部門には ENGINEERING が作成されます。
     ```
     CREATE SERVICE CLASS ENGINEERING
     ```
   - 検査部門には TESTING が作成されます。
     ```
      CREATE SERVICE CLASS TESTING
     ```
   - 生産部門には PRODUCTION が作成されます。
     ```
     CREATE SERVICE CLASS PRODUCTION
     ```
2. Bob は、各部門ごとに適切な許可 ID を持つセッション・ユーザー・グループを作成します。
   - 許可 ID SALESGRP を持つセッション・ユーザー・グループが作成されます。このグループには、販売部門にいるすべてのユーザーの許可 ID が含まれます。
   - 許可 ID ACCTNGRP を持つセッション・ユーザー・グループが作成されます。このグループには、会計部門にいるすべてのユーザーの許可 ID が含まれます。

© Copyright IBM Corp. 2007, 2008

- 許可 ID ENGINGRP を持つセッション・ユーザー・グループが作成されます。このグループには、技術部門にいるすべてのユーザーの許可 ID が含まれます。
- 許可 ID TESTGRP を持つセッション・ユーザー・グループが作成されます。このグループには、検査部門にいるすべてのユーザーの許可 ID が含まれます。
- 許可 ID PRODGRP を持つセッション・ユーザー・グループが作成されます。このグループには、生産部門にいるすべてのユーザーの許可 ID が含まれます。

3. Bob はワークロードを作成し、各グループからの接続を関連したサービス・クラスにマップします。
   - セッション・ユーザー・グループが SALESGRP に設定されたワークロード WL_SALES が作成されます。 WL_SALES は、その接続をサービス・スーパークラス SALES にマップします。

     ```
     CREATE WORKLOAD WL_SALES SESSION_USER GROUP ('SALESGRP')
     SERVICE CLASS SALES
     ```

   - セッション・ユーザー・グループが ACCTNGRP に設定されたワークロード WL_ACCOUNTING が作成されます。 WL_ACCOUNTING は、その接続をサービス・スーパークラス ACCOUNTING にマップします。

     ```
     CREATE WORKLOAD WL_ACCOUNTING SESSION_USER GROUP ('ACCTNGRP')
     SERVICE CLASS ACCOUNTING
     ```

   - セッション・ユーザー・グループが ENGINGRP に設定されたワークロード WL_ENGINEERING が作成されます。 WL_ENGINEERING はその接続をサービス・クラス ENGINEERING にマップします。

     ```
     CREATE WORKLOAD WL_ENGINEERING SESSION_USER GROUP ('ENGINGRP')
     SERVICE CLASS ENGINEERING
     ```

   - セッション・ユーザー・グループが TESTGRP に設定されたワークロード WL_TEST が作成されます。 WL_TEST はその接続をサービス・クラス TESTING にマップします。

     ```
     CREATE WORKLOAD WL_TEST SESSION_USER GROUP ('TESTGRP')
     SERVICE CLASS TESTING
     ```

   - セッション・ユーザー・グループが PRODGRP に設定されたワークロード WL_PRODUCTION が作成されます。 WL_PRODUCTION はその接続をサービス・クラス PRODUCTION にマップします。

     ```
     CREATE WORKLOAD WL_PRODUCTION SESSION_USER GROUP ('PRODGRP')
     SERVICE CLASS PRODUCTION
     ```

Bob は、デフォルトのサービス・クラスおよびワークロード設定を使用します。また、サービス・クラスに対する何らかの制御を行う前に、データベースの使用パターンを監視したいと考えています。結果として作成されるサービス・スーパークラス定義は次のとおりです。

表 17. サービス・クラス定義

サービス・クラス
SALES
ACCOUNTING
ENGINEERING

表 17. サービス・クラス定義 (続き)

TESTING
PRODUCTION
SYSDEFAULTUSERCLASS
SYSDEFAULTMAINTENANCECLASS
SYSDEFAULTSYSTEMCLASS

上記のとおりにインプリメントされたワークロード管理ソリューションを使用すると、各部門からの作業はそれ自体のサービス・スーパークラスにルーティングされます。どの部門からのものかが明確でない作業は、デフォルトのサービス・スーパークラス SYSDEFAULTUSERCLASS にマップされます。この構成を使用して、Bob は各サービス・クラスでの作業をモニターし、部門のデータベース使用パターンを判別することができます。

ワークロード管理インプリメンテーションの最初の改良

最新の接続ピークの後、Bob は WLM_GET_SERVICE_SUPERCLASS_STATS 表関数を使用してサービス・スーパークラスの統計を照会し、各サービス・スーパークラスごとに接続の最高水準点の値を調べます。 Bob は、検査部門を除くすべての部門において接続の最高水準点がほぼ 100 であることに気付きます。ただし、検査部門の統計では、検査チームがある時に 800 を超える接続を確立したことが示されています。

検査部門では、月に一度、月ごとの徹底的な製品テストが行われます。このときに、部門において最大 1000 の同時接続が確立されます。データベース・マネージャー構成パラメーター **max_connections** は 1000 に設定されているため、検査部門が使用可能なデータベースへの接続のほとんどを使用していることになります。システムに 1000 の接続が存在する場合、その後のすべての接続は拒否されます。

システム上のメモリー制約のため、**max_connections** および **maxagents** 構成値をデータ・サーバー上で増やして、より多くの接続を許可することはできません。

Bob は、検査部門がすべての接続を使用してしまわないように、検査部門からの接続数を制限して、他の 4 つの部門がそれぞれのビジネス目標を達成するためにデータベースへの十分な接続を得られるようにすることを決定します。

他の 4 つの部門は通常、それぞれ 150 を超える同時接続を必要としません。さらに、Bob はデフォルト・ユーザー、デフォルト保守、およびデフォルト・システムの各サービス・スーパークラスに接続が含まれることはほとんどないことにも気付きます。それで、これらのデフォルト・サービス・スーパークラスには 100 の接続で十分であると判断します。使用可能な 1000 個の接続を持つ **max_connections** プールから 700 の接続 (600 は 4 つの部門用、100 はデフォルト・クラス用) が割り振られ、検査部門が使用できる接続は 300 になります。検査部門の接続を最大 300 に制限することによって、他の部門からのユーザーは接続要求を拒否されることがなくなります。

検査グループの同時接続を最大 300 に制限するために、Bob は TESTING サービス・クラス用にしきい値が 300 の MAXSERVICECLASSCONNECTIONS を作成します。

```
CREATE THRESHOLD MAXSERVICECLASSCONNECTIONS FOR SERVICE CLASS TESTING ACTIVITIES
ENFORCEMENT DATABASE PARTITION
WHEN TOTALSCPARTITIONCONNECTIONS > 300 STOP EXECUTION
```

この変更を実施した後、ワークロード管理構成は次のようになります。

表 18. TESTING サービス・スーパークラス用のしきい値を追加した後の構成

サービス・クラス	MAXSERVICECLASSCONNECTIONS しきい値
SALES	N/A
ACCOUNTING	N/A
ENGINEERING	N/A
TESTING	300
PRODUCTION	N/A
SYSDEFAULTUSERCLASS	N/A
SYSDEFAULTMAINTENANCECLASS	N/A

TESTING サービス・クラスに含めることができる最大同時接続は 300 のみであるため、このしきい値を上回る接続要求はすべて拒否されます。一方、MAXSERVICECLASSCONNECTIONS しきい値は他のサービス・クラスには適用されないため、これらのサービス・クラスはデータ・サーバーへの残りの 700 個の使用可能な接続を共有します。これらのサービス・クラス間において接続の競合は存在しません。このため、Bob は接続しきい値をそれらのサービス・クラスには設定しません。

ワークロード管理インプリメンテーションの 2 回目の改良

販売、会計、技術、および生産部門からの接続が拒否されることはなくなりましたが、これらの部門のユーザーは依然として、検査部門が徹底的な製品検査を行っているときにパフォーマンスが低下することについて苦情を述べています。 Bob は、検査部門がその製品検査サイクル中に実行する照会を調べ、その照会に大量のデータが関係する複雑な結合が含まれていることに気付きます。これらの照会では、非常に多くのプリフェッチ・アクティビティーが生成されます。これにより、他の部門からの接続のプリフェッチ要求が処理できなくなります。 Bob は検査部門からの接続のプリフェッチ優先順位を下げることにし、TESTING サービス・クラスのプリフェッチ優先順位の設定を LOW に変更します。

```
ALTER SERVICE CLASS TESTING PREFETCH PRIORITY LOW
```

ワークロード管理構成は次のようになります。

表 19. TESTING サービス・スーパークラス用のプリフェッチ優先順位を変更した後の構成

サービス・クラス	MAXSERVICECLASSCONNECTIONS しきい値	プリフェッチの優先順位
SALES	N/A	DEFAULT
ACCOUNTING	N/A	DEFAULT
ENGINEERING	N/A	DEFAULT
TESTING	300	LOW
PRODUCTION	N/A	DEFAULT

表 19. TESTING サービス・スーパークラス用のプリフェッチ優先順位を変更した後の構成 (続き)

サービス・クラス	MAXSERVICECLASSCONNECTIONS しきい値	プリフェッチの優先順位
SYSDEFAULTUSERCLASS	N/A	DEFAULT
SYSDEFAULTMAINTENANCECLASS	N/A	DEFAULT

TESTING サービス・クラスのプリフェッチ優先順位を LOW に設定すると、検査部門から発行された接続からのプリフェッチ要求は、他の部門からのすべてのプリフェッチ要求が処理された後に初めて扱われるようになります。この変更により、他の部門の照会スループットは増加し、検査部門の製品検査フェーズ中のスループットは低下します。

ワークロード管理インプリメンテーションの 3 回目の改良

プリフェッチ問題が解決した後、技術部門は Brewmeister という実験的なアプリケーション用にいくつかの接続を必要としていることを Bob に伝えます。そのアプリケーションは実験用であるため、Bob はそれが多くのデータベース接続を消費しないようにするとともに、システムがビジーである場合はそのアプリケーションからの照会がプリフェッチャーの競合を起こさないようにしたいと思っています。これらの目標を達成するため、実験的アプリケーション用の新規のサービス・サブクラスを ENGINEERING サービス・スーパークラスの下に作成し、さらにアプリケーションからの接続を新規のサービス・サブクラスにマップするためのワークロードを作成します。 Bob はサービス・クラスおよびワークロードを次のように更新します。

- サービス・サブクラス EXPERIMENT がサービス・スーパークラス ENGINEERING の下に作成されます。

  ```
  CREATE SERVICE CLASS EXPERIMENT UNDER ENGINEERING
  ```

- しきい値が 50 の MAXSERVICECLASSCONNECTIONS がサービス・サブクラス EXPERIMENT 用に作成されます。

  ```
  CREATE THRESHOLD MAXSERVICECLASSCONNECTIONS FOR SERVICE CLASS EXPERIMENT
  UNDER ENGINEERING ACTIVITIES
  ENFORCEMENT DATABASE WHEN TOTALDBPARTITIONCONNECTIONS > 50 STOP EXECUTION
  ```

- アプリケーション BREWMEISTER からの接続をサービス・サブクラス EXPERIMENT にマップするためにワークロード WL_EXPERIMENT が作成されます。

  ```
  CREATE WORKLOAD WL_EXPERIMENT APPLNAME ('BREWMEISTER') SERVICE CLASS EXPERIMENT
  UNDER ENGINEERING
  ```

- EXPERIMENT サービス・サブクラスのプリフェッチ優先順位は LOW に設定されます。

  ```
  ALTER SERVICE CLASS EXPERIMENT UNDER ENGINEERING PREFETCH PRIORITY LOW
  ```

ワークロード管理構成は次のようになります。

表 20. EXPERIMENT サービス・サブクラスを使用した構成

サービス・クラス	MAXSERVICECLASSCONNECTIONS しきい値	プリフェッチの優先順位
SALES	N/A	DEFAULT

表 20. EXPERIMENT サービス・サブクラスを使用した構成 (続き)

サービス・クラス	MAXSERVICECLASSCONNECTIONS しきい値	プリフェッチの優先順位
ACCOUNTING	N/A	DEFAULT
ENGINEERING	N/A	DEFAULT
EXPERIMENT	50	LOW
TESTING	300	LOW
PRODUCTION	N/A	DEFAULT
SYSDEFAULTUSERCLASS	N/A	DEFAULT
SYSDEFAULTMAINTENANCECLASS	N/A	DEFAULT

この構成により、BREWMEISTER アプリケーションは 50 個のデータベースへの同時接続のみを保持することができます。さらに、このアプリケーションからのプリフェッチ要求は、優先順位が低いプリフェッチ・キューに送信されます。これで、誤ってデータベース・システムの処理を不能にしてしまうことはないとわかり、技術部門はアプリケーションを使用した実験を安全に行うことができるようになりました。

例: ワークロードの割り当て

データベース接続が確立された後の最初の作業単位の初めに、使用可能になっている各ワークロードの接続属性を評価することによって、データ・サーバーは接続をワークロードに割り当てます。接続属性の値またはワークロード定義自体が作業単位の間に変更される場合には、各作業単位の初めにワークロードの再評価が行われます。

以下の図は、ワークロードの割り当てを示しています。`AppA` を介して照会をサブミットする `Marketing` グループのユーザーは、APPAQUERIES ワークロードに割り当てられます。PAYROLL が APPAQUERIES の前に置かれているにも関わらず、PAYROLL ワークロードには割り当てられません。これは、ワークロード PAYROLL の定義が SESSION_USER GROUP キーワードを `Finance` として指定しているためです。`AppA` を介して照会をサブミットする `Finance` グループのユーザーは、FINANCE ワークロードに割り当てられます。PAYROLL ワークロードの方がより特定的で、かつその定義に `AppA` と `Finance` の両方が指定されているにも関わらず、PAYROLL ワークロードには割り当てられません。これは、FINANCE ワークロードが PAYROLL ワークロードの前に置かれているためです。`AppB` を介して照会をサブミットする `Marketing` グループのユーザーは、SYSDEFAULTUSERWORKLOAD ワークロードに割り当てられます。これは、FINANCE、PAYROLL、または APPAQUERIES ワークロード定義に指定されている接続属性がどれも `AppB` アプリケーションまたは `Marketing` グループと一致しないためです。

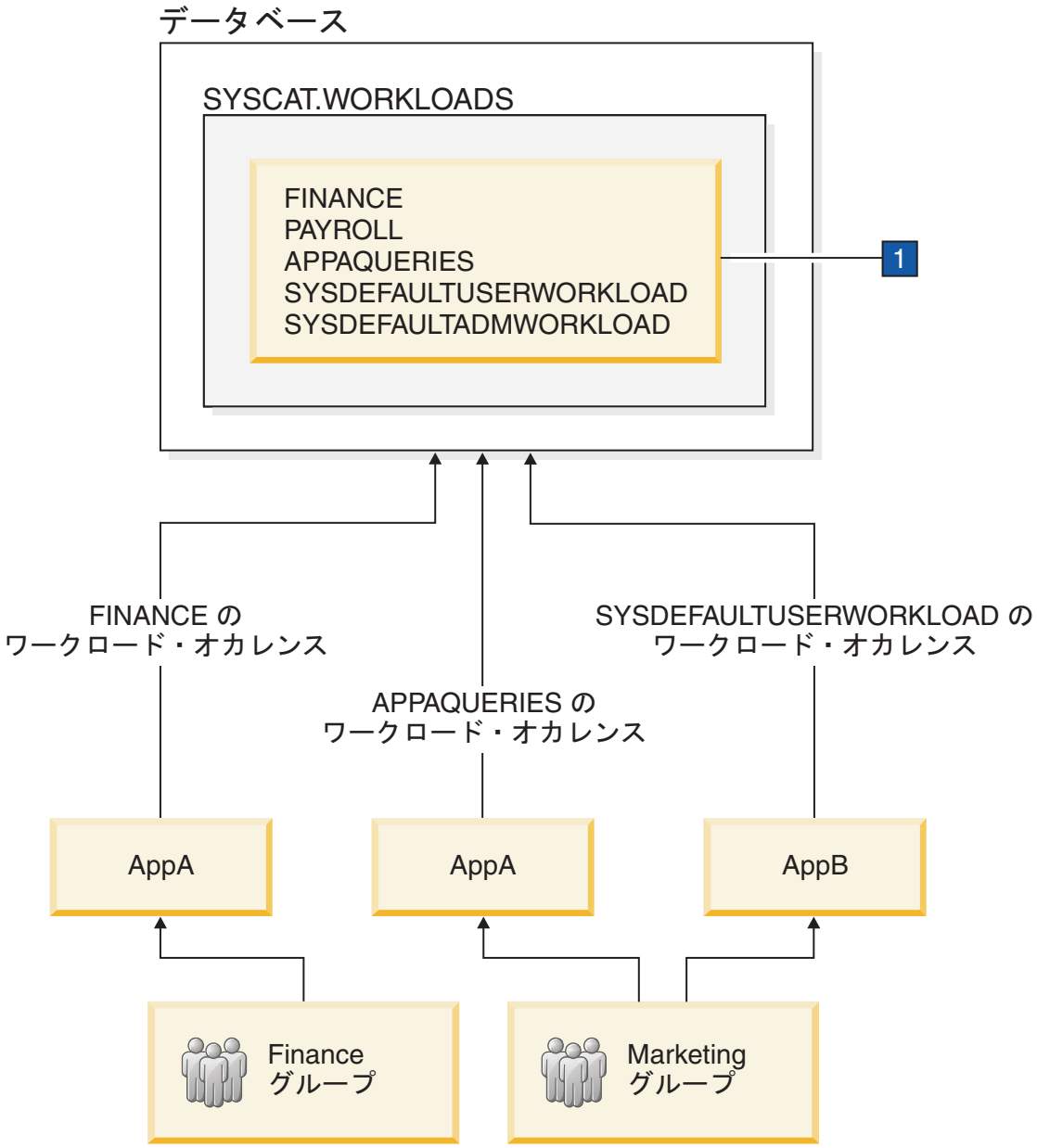


図 25. ワークロードの割り当ての例

1 前の図では、CREATE WORKLOAD ステートメントは次のとおりです。

```
CREATE WORKLOAD PAYROLL APPLNAME ('AppA') SESSION_USER GROUP ('FINANCE')
SERVICE CLASS SC1
CREATE WORKLOAD APPAQUERIES APPLNAME('AppA') POSITION LAST
SERVICE CLASS SC2

CREATE WORKLOAD FINANCE SESSION_USER GROUP ('FINANCE') SERVICE CLASS SC1
POSITION BEFORE PAYROLL
```

3 層から成るクライアント/サーバー環境では、データベース接続は、アプリケーションサーバーから確立されます。アプリケーション・サーバーは sqleseti (クライアント情報の設定) API を使用して、クライアント情報を DB2 データ・サーバーに渡すことができます。そうでない場合は、アプリケーション・サーバーからの情報のみが渡され、その情報は、このアプリケーション・サーバーを介して送付される

すべてのクライアント要求で同じになると思われます。データ・サーバーは、異なるクライアントからの作業単位を異なるワークロード (および異なるサービス・クラス) に割り当てる場合、作業単位とワークロードを関連付けるための基準として、クライアント情報属性 (つまり、クライアント・ユーザー ID、クライアント・アプリケーション名、クライアント・ワークステーション名、およびクライアント会計情報ストリング) を使用します。

以下の図は、セッション・ユーザー APPUSER を使用してデータベースへの接続を確立するアプリケーション・サーバーを介して、さまざまなユーザー・アプリケーション (marketing.exe、auditing.exe、および reporting.exe) が照会をサブミットする、3 層の環境の例を示しています。次の 3 つのワークロードが定義されます。marketing.exe によってサブミットされる照会用のワークロード、reporting.exe によってサブミットされる照会用のワークロード、および残りの照会用のワークロードです。図に示されているとおり、marketing.exe によってサブミットされる照会を MARKETING ワークロードに割り当てるには、アプリケーション・サーバーは sqleseti API を呼び出して、CURRENT CLIENT_APPLNAME 特殊レジスターの値を marketing.exe に設定します。同様に、reporting.exe によってサブミットされる照会を REPORTING ワークロードに割り当てるには、サーバーは sqleseti を呼び出して、CURRENT CLIENT_APPLNAME 特殊レジスターの値を reporting.exe に設定します。図でサーバーが sqleseti を呼び出して CURRENT CLIENT_USERID 特殊レジスターを Lidia に設定する (他には何も変更されない。つまりクライアント・アプリケーション名が引き続き reporting.exe に設定される) 場合、ワークロードの再割り当ては生じないということに注意してください。これは、特に CURRENT CLIENT_USERID が Lidia に設定されて定義されているワークロードがないためです。

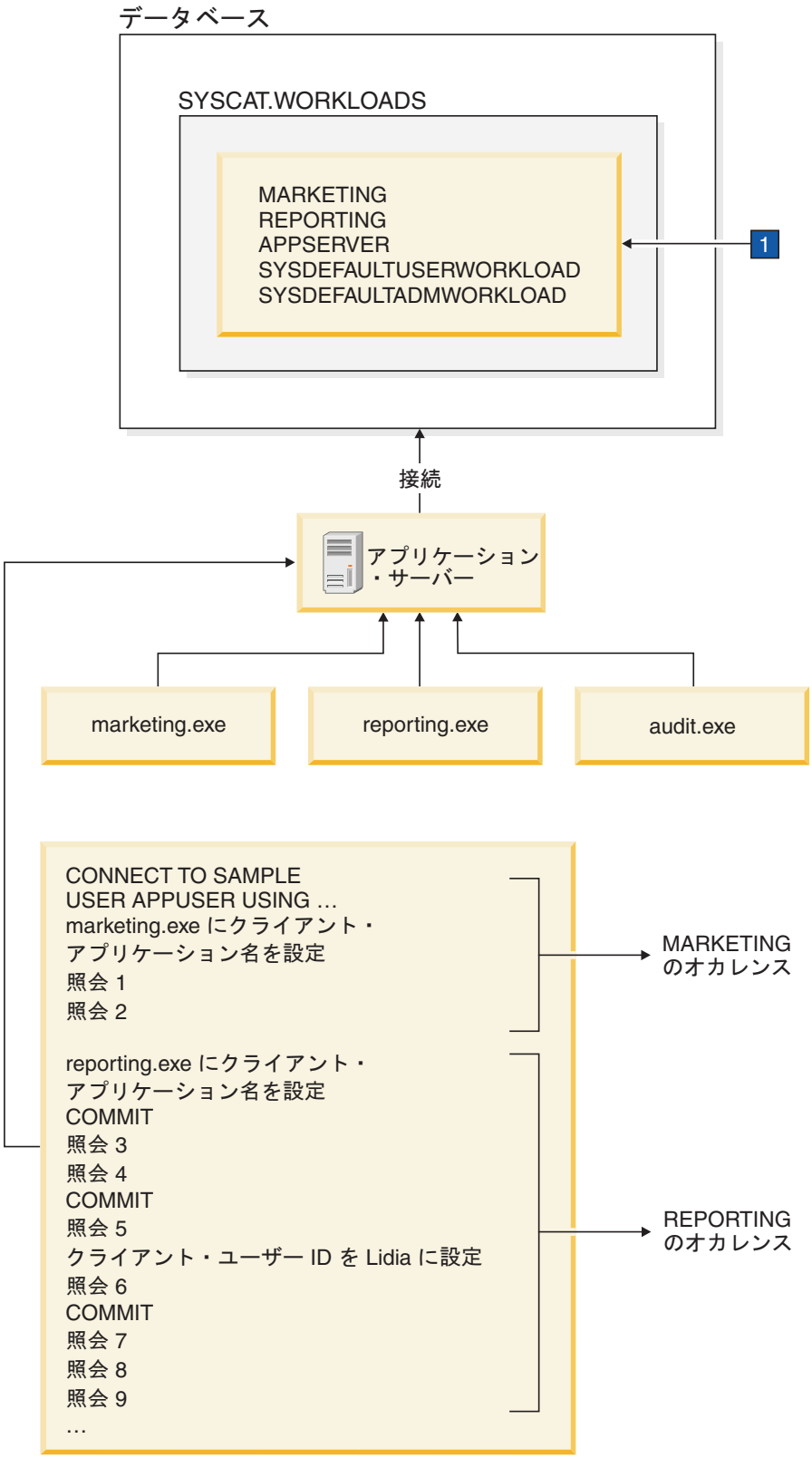


図 26. 3 層環境におけるワークロードの割り当ての例

以下のステートメントは、前の図の枠 **1** で指定されたワークロードを定義するために使用されます。

```
CREATE WORKLOAD MARKETING SESSION_USER ('APPUSER')
CURRENT CLIENT_APPLNAME ('marketing.exe') SERVICE CLASS SC2
POSITION AT 1

CREATE WORKLOAD REPORTING SESSION USER ('APPUSER')
CURRENT CLIENT_APPLNAME ('reporting.exe') SERVICE CLASS SC4
POSITION AFTER MARKETING

CREATE WORKLOAD APPSERV SESSION_USER ('APPUSER')
SERVICE CLASS SC1
```

例: ワークロード属性に単一値が含まれる場合のワークロードの割り当て

このトピックの例では、データ・サーバーがワークロードの割り当てを実行する方法について示します。この例では、各ワークロード接続属性ごとに 1 つの値のみが指定されます。

以下のワークロードがカタログ内に存在すると想定します。

表 21. カタログ内のワークロード

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
1	REPORTS	AppA								
2	INVENTORY REPORT	AppB	LYNN		ACCOUNTING	TELEMKTR				
3	SALES REPORT	AppC	KATE	KATE		SALESREP				
4	AUDIT REPORT	AppB			ACCOUNTING	FINANALYST				
5	EXPENSE REPORT	AppA	TIM			EXPENSE APPROVER				
6	AUDIT RESULT			LYNN			LYNN			Audit Group

以下の属性を持つデータベース接続が確立されると想定します。

表 22. データベース接続属性

APPLNAME	SYSTEM_USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
AppA	TIM	TIM	FINANCE	FINANALYST、EXPENSE APPROVER	NULL	NULL	NULL	Business account

最初の作業単位がサブミットされるときに、データ・サーバーは、リストにある最初のワークロードから始めて、カタログ内の各ワークロードを検査します。さらに、一致する属性を含むワークロードが見つかるまで、ワークロードを昇順で処理します。一致するワークロードが見つかると、作業単位はそのワークロードのオカレンスの下で実行されます。接続を割り当てるワークロードを決定する場合、データ・サーバーは接続属性を決定論的順序で比較します。

データ・サーバーは最初に REPORTS ワークロードを検査して一致がないかどうか調べます。REPORTS ワークロードはリストの最初にあります。

表 23. カタログ内の REPORTS ワークロード

評価順序	ワークロード名	APPLNAME	SYSTEM_USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
1	REPORTS	AppA								

データ・サーバーは、以下の決定論的順序で接続属性を検査します。

1. APPLNAME。データベース接続の APPLNAME の値 AppA は REPORTS ワークロード用の APPLNAME の値と一致します。
2. SYSTEM_USER。これはワークロード定義では設定されていません。すべての値 (NULL 値を含む) が一致であると見なされます。
3. SESSION_USER。これはワークロード定義では設定されていません。すべての値が一致であると見なされます。
4. SESSION_USER GROUP。これはワークロード定義では設定されていません。すべての値が一致であると見なされます。
5. SESSION_USER ROLE。これはワークロード定義では設定されていません。すべての値が一致であると見なされます。
6. CURRENT CLIENT_USERID。これはワークロード定義では設定されていません。すべての値が一致であると見なされます。
7. CURRENT CLIENT_APPLNAME。これはワークロード定義では設定されていません。すべての値が一致であると見なされます。
8. CURRENT CLIENT_WRKSTNNAME。これはワークロード定義では設定されていません。すべての値が一致であると見なされます。
9. CURRENT CLIENT_ACCTNG。これはワークロード定義では設定されていません。すべての値が一致であると見なされます。

この場合、REPORTS ワークロードの接続属性と接続で渡される情報との間に明示的および暗黙的な一致が存在するために、データ・サーバーは REPORTS ワークロードを潜在的な一致として選択します。ワークロードを選択した後、データ・サーバーは、セッション・ユーザーにそのワークロードに対する USAGE 特権があるかどうかを検査します。セッション・ユーザー TIM に REPORTS ワークロードに対する USAGE 特権がある場合、そのワークロードは接続に使用されます。ただし、TIM が REPORTS ワークロードに対する USAGE 特権を所有しない場合、データ・サーバーは、INVENTORYREPORT ワークロードを検査して一致がないかどうかを調べることによって処理を続行します。

EXPENSEREPORT ワークロードで追加の接続属性が指定されているため、TIM をそのワークロードに割り当てたいと仮定します。この場合、ワークロードの評価順序を変更して、次のようにワークロード・リストの REPORTS の前に EXPENSEREPORT を配置します。

```
ALTER WORKLOAD EXPENSEREPORT POSITION AT 1
```

さらに、以下の SQL ステートメントを使用して同じ結果を得ることもできます。

```
ALTER WORKLOAD EXPENSEREPORT BEFORE REPORTS
```

ALTER WORKLOAD ステートメントを有効にするには、COMMIT ステートメントを ALTER WORKLOAD ステートメントの直後に発行する必要があります。カタログでの ALTER WORKLOAD ステートメントの効果は次のとおりです。

表 24. EXPENSEREPORT ワークロードを位置変更した後のカタログ内のワークロード

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
1	EXPENSE REPORT	AppA	TIM			EXPENSE APPROVER				
2	REPORTS	AppA								

表 24. *EXPENSEREPORT ワークロードを位置変更した後のカタログ内のワークロード (続き)*

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
3	INVENTORY REPORT	AppB	LYNN		ACCOUNTING	TELEMKTR				
4	SALES REPORT	AppC	KATE	KATE		SALESREP				
5	AUDIT REPORT	AppB			ACCOUNTING	FINANALYST				
6	AUDIT RESULT			LYNN			LYNN			Audit Group

TIM がまだ EXPENSEREPORT ワークロードに対する USAGE 特権を持っていない場合は、以下のステートメントを発行する必要があります (COMMIT ステートメントによって GRANT ステートメントを有効にします)。

```
GRANT USAGE ON WORKLOAD EXPENSEREPORT TO USER TIM
COMMIT
```

次の作業単位の開始時に、ワークロードの再割り当てが実行され、データ・サーバーは TIM からの接続を EXPENSEREPORT ワークロードに割り当てます。さらに、同じ属性を持つ他の接続によってサブミットされた新しい作業単位も、EXPENSEREPORT ワークロードと関連付けられます。

例: 複数のワークロードが存在する場合の作業単位に対するワークロードの割り当て

このトピックの例では、データ・サーバーが既存のワークロードへの接続を割り当てるために、ワークロード評価を実行する方法を示します。

以下のワークロードがカタログ内で定義されていると想定します。

表 25. *カタログ内のワークロード*

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
1	EXPENSE REPORT	AppB	TIM			EXPENSE APPROVER				
2	REPORTS	AppB								
3	INVENTORYREPORT	AppA	LYNN		ACCOUNTING	TELEMKTR				
4	SALES REPORT	AppC	KATE	KATE		SALESREP				
5	AUDIT REPORT	AppA			ACCOUNTING	FINANALYST				
6	AUDIT RESULT			LYNN			LYNN			Audit Group

以下の属性を持つデータベース接続が確立されると想定します。

表 26. *データベース接続属性*

APPLNAME	SYSTEM_USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
AppA	LYNN	LYNN	ACCOUNTING	FINANALYST, SALESREP	LYNN	NULL	wrkstn2	Audit group

最初の作業単位がサブミットされると、データ・サーバーはカタログ内の各ワークロードを昇順の評価順序で検査し、接続が提供するものと接続属性が一致するワークロードを見つけると停止します。ワークロードを確認する際、データ・サーバーは接続属性を決定論的順序で比較します。

最初に、データ・サーバーは EXPENSEREPORT ワークロードを検査します。

表 27. カタログ内の EXPENSEREPORT ワークロード

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
1	EXPENSEREPORT	AppB	TIM			EXPENSE APPROVER				

ワークロード定義にある APPLNAME 属性は AppB ですが、接続から渡された APPLNAME 属性は AppA なので、マッチングは不可能です。データ・サーバーは、リストの 2 番目の REPORTS ワークロードへ進みます。

表 28. カタログ内の REPORTS ワークロード

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
2	REPORTS	AppB								

この場合もやはり、ワークロード定義にある APPLNAME 属性は AppB なので AppA と一致しません。データ・サーバーはリスト内の 3 番目のワークロード、INVENTORYREPORT へ進みます。

表 29. カタログ内の INVENTORYREPORT ワークロード

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
3	INVENTORYREPORT	AppA	LYNN		ACCOUNTING	TELEMKTR				

データ・サーバーはサブミットされた接続属性と INVENTORYREPORT ワークロードの間の一致を検査します。属性が以下の順で検査されます

1. APPLNAME。ワークロード定義および接続の両方が AppA の値を持つため、一致します。
2. SYSTEM_USER。ワークロード定義および接続の両方が LYNN の値を持つため、一致します。
3. SESSION_USER。接続は LYNN の値を渡しました。ワークロードに SESSION_USER 属性は設定されていないため、NULL 値を含め接続から渡されたどんな値でも一致します。
4. SESSION_USER GROUP。ワークロード定義および接続の両方が ACCOUNTING の値を持つため、一致します。
5. SESSION_USER ROLE。ワークロード定義は TELEMKTR の値を指定していますが、接続は FINANALYST および SALESREP の値を提供しました。この属性には一致が発生しません。

データ・サーバーは INVENTORYREPORT ワークロードと接続の属性を突き合わせようとするのを停止し、リストにある 4 番目のワークロードである SALESREPORT へ進みます。

表 30. カタログ内の SALESREPORT ワークロード

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
4	SALESREPORT	AppC	KATE	KATE		SALESREP				

SALESREPORT ワークロード定義の APPLNAME が AppC なので、接続 (APPLNAME に対して AppA の値を渡した) との一致は発生しません。そこで、データ・サーバーはリスト内の 5 番目のワークロード、AUDITREPORT へ進みます。

表 31. カタログ内の AUDITREPORT ワークロード

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
5	AUDITREPORT	AppA			ACCOUNTING	FINANALYST				

データ・サーバーは、AUDITREPORT ワークロードと接続の属性を決定論的順序で比較します。

1. APPLNAME。ワークロード定義および接続の両方が AppA の値を持つため、一致します。
2. SYSTEM_USER。接続は LYNN の値を渡しました。ワークロードに SYSTEM_USER 属性は設定されていないため、接続から渡されたどんな値でも一致します。
3. SESSION_USER。接続は LYNN の値を渡しました。ワークロードに SESSION_USER 属性は設定されていないため、接続から渡されたどんな値でも一致します。
4. SESSION_USER GROUP。ワークロードおよび接続の両方がこの属性に対して ACCOUNTING の値を持つため、一致します。
5. SESSION_USER ROLE。ワークロードおよび接続の両方がこの属性に対して FINANALYST の値を持つため、一致します。
6. CURRENT CLIENT_USERID。ワークロードに CURRENT CLIENT_USERID 属性は設定されていないため、接続から渡されたどんな値でも一致します。
7. CURRENT CLIENT_APPLNAME。ワークロードに CURRENT CLIENT_APPLNAME 属性は設定されていないため、接続から渡されたどんな値でも一致します。
8. CURRENT CLIENT_WRKSTNNAME。ワークロードに CURRENT CLIENT_WRKSTNNAME 属性は設定されていないため、接続から渡されたどんな値でも一致します。
9. CURRENT CLIENT_ACCTNG。ワークロードに CURRENT CLIENT_ACCTNG 属性は設定されていないため、接続から渡されたどんな値でも一致します。

すべての接続属性を処理し、一致するワークロードを見つけた後、データ・サーバーはセッション・ユーザーにそのワークロードに対する USAGE 特権があるかどうかを検査します。LYNN は AUDITREPORT ワークロードに対する USAGE 特権を持っていないと想定します。この状態では、たとえすべての接続属性が一致しても、このワークロードは接続と関連付けられません。データ・サーバーは評価リスト内の 6 番目のワークロード、AUDITRESULT へ進みます。

表 32. カタログ内の AUDITRESULT ワークロード

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURRENT CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
6	AUDITRESULT			LYNN			LYNN			Audit Group

データ・サーバーは、AUDITRESULT ワークロードと接続の属性を決定論的順序で比較します。

1. APPLNAME。ワークロードに APPLNAME 属性は設定されていないため、接続から渡されたどんな値でも一致します。
2. SYSTEM_USER。ワークロードに SYSTEM_USER 属性は設定されていないため、接続から渡されたどんな値でも一致します。
3. SESSION_USER。ワークロードおよび接続の両方がこの属性に対して LYNN の値を持つため、一致します。
4. SESSION_USER GROUP。ワークロードに SESSION_USER GROUP 属性は設定されていないため、接続から渡されたどんな値でも一致します。
5. SESSION_USER ROLE。ワークロードに SESSION_USER ROLE 属性は設定されていないため、接続から渡されたどんな値でも一致します。
6. CURRENT CLIENT_USERID。ワークロードおよび接続の両方がこの属性に対して LYNN の値を持つため、一致します。
7. CURRENT CLIENT_APPLNAME。ワークロードに CURRENT CLIENT_APPLNAME 属性は設定されていないため、接続から渡されたどんな値でも一致します。
8. CURRENT CLIENT_WRKSTNNAME。ワークロードに CURRENT CLIENT_WRKSTNNAME 属性は設定されていないため、接続から渡されたどんな値でも一致します。
9. CURRENT CLIENT_ACCTNG。ワークロードおよび接続の両方がこの属性に対して Audit Group の値を持つため、一致します。

すべての接続属性を処理し、一致するワークロードを見つけた後、データ・サーバーはセッション・ユーザーにそのワークロードに対する USAGE 特権があるかどうかを検査します。この状態で、セッション・ユーザー LYNN は AUDITRESULT ワークロードに対する USAGE 特権を持つと想定します。接続属性がすべて一致し、セッション・ユーザーが USAGE 特権を持っているので、接続は AUDITRESULT ワークロードに割り当てられます。

例: ワークロード属性に複数の値が含まれる場合のワークロードの割り当て

このトピックの例では、データ・サーバーがワークロードの割り当てを実行する方法について示します。この例では、いくつかのワークロード定義は 1 つの接続属性に対して複数の値を許可します。

以下のワークロードがカタログ内で定義されていると想定します。

表 33. カタログ内のワークロード

評価順序	ワークロード名前	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURREN CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
1	ITEMINQ		KYLE, GEORGE		RETAIL, SALES					
2	DAILY TRANS REPORT	AppC		KYLE, CAROL	SALES, ACCOUNTING					
3	SALES SUMMARY	AppA, AppB				ACCOUNTANT, FINANALYST				

以下の属性を持つデータベース接続が確立されると想定します。

表 34. データベース接続属性

APPLNAME	SYSTEM_USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURREN CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
AppC	LINDA	KYLE	SALES	ACCOUNTANT	LINDA	NULL	NULL	Business Account

最初の作業単位がサブミットされると、データ・サーバーはカタログ内の各ワークロードを昇順の評価順序で検査し、接続が提供するものと接続属性が一致するワークロードを見つけると停止します。ワークロードを確認する際、データ・サーバーは接続属性を決定論的順序で比較します。

最初に、データ・サーバーは ITEMINQ ワークロードを検査します。

表 35. カタログ内の ITEMINQ ワークロード

評価順序	ワークロード名	APPLNAME	SYSTEM_USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURREN CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
1	ITEMINQ		KYLE, GEORGE		RETAIL, SALES					

データ・サーバーはサブミットされた接続属性と ITEMINQ ワークロードの間の一致を検査します。属性が以下の順で検査されます

1. APPLNAME。ワークロードに APPLNAME 属性は設定されていないため、NULL 値を含め、接続から渡されたどんな値でも一致します。
2. SYSTEM_USER。接続は LINDA の値を渡しました。しかし、ITEMNO ワークロードの値は KYLE および GEORGE です。この属性には一致が発生しません。

データ・サーバーは ITEMNO ワークロードと接続を突き合わせようとするのを停止し、リストにある 2 番目のワークロードである DAILYTRANSREPORT へ進みます。

表 36. カタログ内の DAILYTRANSREPORT ワークロード

評価順序	ワークロード名	APPLNAME	SYSTEM _USER	SESSION _USER	SESSION _USER GROUP	SESSION _USER ROLE	CURRENT CLIENT _USERID	CURRENT CLIENT _APPLNAME	CURREN CLIENT _WRKSTNNAME	CURRENT CLIENT _ACCTNG
2	DAILYTRANSREPORT	AppC		KYLE, CAROL	SALES, ACCOUNTING					

データ・サーバーは、DAILYTRANSREPORT ワークロードと接続の属性を決定論的順序で比較します。

1. APPLNAME。ワークロード定義および接続の両方が AppC の値を持つため、一致します。
2. SYSTEM_USER。ワークロードに SYSTEM_USER 属性は設定されていないため、NULL 値を含め、接続から渡されたどんな値でも一致します。
3. SESSION_USER。接続によって渡された SESSION_USER の値は KYLE で、ワークロード SESSION_USER の値の 1 つと一致します。接続から渡されたのが CAROL だったとしても、KYLE と CAROL の両方が DAILYTRANSREPORT ワークロード定義の一部として指定されているので、一致することになります。
4. SESSION_USER GROUP。接続によって渡された SESSION_USER GROUP の値は SALES で、ワークロード SESSION_USER GROUP の属性に対して指定された SALES の値に一致します。接続から渡されたのが ACCOUNTING だったとしても、SALES と ACCOUNTING の両方がワークロード定義に指定されているので、一致することになります。

5. SESSION_USER ROLE。ワークロードに SESSION_USER ROLE 属性は設定されていないため、接続から渡されたどんな値でも一致します。
6. CURRENT CLIENT_USERID。ワークロードに CURRENT CLIENT_USERID 属性は設定されていないため、接続から渡されたどんな値でも一致します。
7. CURRENT CLIENT_APPLNAME。ワークロードに CURRENT CLIENT_APPLNAME 属性は設定されていないため、接続から渡されたどんな値でも一致します。
8. CURRENT CLIENT_WRKSTNNAME。ワークロードに CURRENT CLIENT_WRKSTNNAME 属性は設定されていないため、接続から渡されたどんな値でも一致します。
9. CURRENT CLIENT_ACCTNG。ワークロードに CURRENT CLIENT_WRKSTNNAME 属性は設定されていないため、接続から渡されたどんな値でも一致します。

すべての接続属性を処理し、接続に一致するワークロードを見つけた後、データ・サーバーはセッション・ユーザーにそのワークロードに対する USAGE 特権があるかどうかを検査します。この状態で、セッション・ユーザー KYLE は DAILYTRANSREPORT ワークロードに対する USAGE 特権を持つと想定します。接続属性がすべて一致し、セッション・ユーザーが USAGE 特権を持っているので、接続は DAILYTRANSREPORT ワークロードに割り当てられます。

例: しきい値の使用

しきい値をさまざまな目的に使用することができます。このトピックのシナリオでは、しきい値を使用して、大規模なジョブの実行数を制御し、さまざまなアプリケーションに異なる実行時間を許可し、開発中のアプリケーションの動作を制御します。

ワークロード管理ソリューションを設定する 1 つの方法は、社内のさまざまな部署に渡ってデータベースのリソースを分割し、管理することです。例えば、営業部門が 2 つのメイン・レポートを管理しており、そこには月間および年間の売り上げが含まれていると想定します。また、人事部門が 1 週間おきに給与計算のアプリケーションを実行し、開発チームは管理チームの要請で新しいタイプのレポートを作成中であると想定します。

この状態で、これらのアプリケーションの 1 つ 1 つに対してワークロード定義を作成し、そのアプリケーションを適用可能なサービス・スーパークラスへマップします。その結果、データベース・カタログには以下のワークロード定義が含まれます。

- MonthlySales (サービス・スーパークラス Sales へマッピングする)
- YearlySales (サービス・スーパークラス Sales へマッピングする)
- Payroll (サービス・スーパークラス Human Resources へマッピングする)
- NewReport (サービス・スーパークラス Development へマッピングする)

大規模なジョブの数のしきい値

YearlySales レポートは非常に大規模なので、データベース内でこのアプリケーションの複数のオカレンスが実行されている、という状況を常に避けたいと思います。そこで、しきい値を作成して、このワークロードの並行オカレンスの最大数を 1 に設定します。

類似の解決策は、YearlySales アプリケーションをサービス・サブクラス YearlySalesReports (Sales サービス・スーパークラスの下にある) に関連付け、またサービス・サブクラスの最大の並行性しきい値を値 1 に設定することによって実現できます。

どちらの状態でも、しきい値のアクションを STOP EXECUTION に設定して、ワークロードの複数のオカレンスが実行されないようにすることができます。しきい値に違反するときの条件についての追加情報を知りたい場合は、アクティビティー情報を収集することもできます。

アクティビティーの存続時間のしきい値

全アプリケーションは 1 時間以下の時間で実行を完了することが期待されるので、データベース・ドメインを使用してしきい値を作成し、どのアクティビティーも 1 時間を超えて実行することを禁止します。この規則の唯一の例外は、完了するのに最大 5 時間かかる年間のレポートだけです。それで、新規のサービス・サブクラスを Sales サービス・スーパークラスの下に作成して、それを YearlySalesReports と名付けます。それから YearlySales ワークロードをこのサービス・サブクラスにマップし、5 時間というアクティビティーの合計時間しきい値を作成し、このしきい値をサービス・サブクラスに関連付けます。データベースの他の部分では 1 時間というグローバルな値が適用されますが、これにより、5 時間という新しい値が YearlySalesReports サービス・サブクラスに適用されるようになります。

コーディネーターの数およびネストされたアクティビティーのしきい値

NewReport アプリケーションはストアード・プロシージャーおよびユーザー定義関数を頻繁に使用し、まだ完全にデバッグされていないため、システムの残りの部分に影響を与えるアクティビティーを多数生成する傾向があります。開発者に相談すると、この新規のレポートは合計して 20 を超えるアクティビティーを生成しないことになっている、ということが分かります。それで、NewReport ワークロード上にワークロード・アクティビティー・タイプのしきい値を定義して、それを 20 に設定します。最初は、しきい値のアクションを STOP EXECUTION および COLLECT ALL に設定して、意に反するアプリケーションの副次作用で多数のアクティビティーが開始しないようにするとともに、開発者が問題を識別しやすくなるようにします。

アプリケーションがより安定したなら、今度は最適化のフェーズに入ります。このフェーズ中に、開発者はアプリケーションが生成するアクティビティーの数を、15 から 20 という数から、15 に削減しようとします。この時、しきい値を変更して、上限の値を 15 に、しきい値のアクションを CONTINUE にします。このしきい値の定義は、生成されたアクティビティーの数が 15 を超えた状態を識別して、それ

に対応するのに役立ちますが、アプリケーションの安定度が増したため、実行を停止する必要はありません。

例: CONCURRENTWORKLOADOCCURRENCES、TOTALDBPARTITIONCONNECTIONS、および TOTALSCPARTITIONCONNECTIONS しきい値

このトピックの例では、CONCURRENTWORKLOADOCCURRENCES、TOTALDBPARTITIONCONNECTIONS、および TOTALSCPARTITIONCONNECTIONS 集約しきい値が相互作用する方法を示します。

データベース接続がデータベース・パーティション 1 上で確立されるケースを想定します。TOTALDBPARTITIONCONNECTIONS しきい値はこの接続がデータベースに接続できるかどうかを決定するため、1 回評価されます。このしきい値の評価が成功すると、TOTALDBPARTITIONCONNECTIONS は同じ接続に対して (それはもうデータベース内にあるので) 二度と評価されません。関連する接続属性が変更されたので、接続はワークロード A とワークロード B との間で何度か切り替えられます。接続がワークロードを切り替えるたびに、接続は別のサービス・クラスへ移動することもあり得ます。いずれかのワークロードのオカレンスの開始時に、CONCURRENTWORKLOADOCCURRENCES しきい値は評価されます。このしきい値に違反していなければ、接続は対応するサービス・クラスに加わります。この時点で、最後のしきい値である TOTALSCPARTITIONCONNECTIONS が評価されます。この時点で、接続はサービス・クラスに入って何らかの作業を発行できるようになる前に、キューに入れられる可能性があります。

例: 特定のアクティビティー・タイプを管理するためのワーク・クラス・セットの使用

以下の例では、ワーク・クラス・セットを使用して CALL および DML アクティビティーを管理する方法について示します。

毎日非常に多くのアプリケーションを NONAME データベースで実行しており、最近少しのパフォーマンス問題が発生しているとします。これらの問題の一部に対応するには、データベースで同時に実行可能な大規模な照会 (つまり、見積コストが 9999 timeron より大きい、または見積カーディナリティーが 9999 行より大きいすべての照会) の数を制御する必要があると判断します。さらに、MYSCHEMA スキーマの任意のルーチンを呼び出す CALL ステートメントの数が 5 を超えた場合に通知を受けるようにしたいとします。

データベースに対して実行可能な大規模な照会および CALL ステートメントの数を制御するには、以下のようにします。

1. 次の 3 つのワーク・クラスが入っている MYWORKCLASSSET ワーク・クラス・セットを作成します。見積コストが大きい照会用、見積カーディナリティーが大きい照会用、および MYSCHEMA スキーマ内のプロシージャーを呼び出す CALL ステートメント用のワーク・クラスです。以下に例を示します。

```
CREATE WORK CLASS SET MYWORKCLASSSET
 (WORK CLASS LARGEESTIMATEDCOST WORK TYPE DML
FOR TIMERONCOST FROM 10000 TO UNBOUNDED,
WORK CLASS LARGECARDINALITY WORK TYPE DML
FOR CARDINALITY FROM 10000 TO UNBOUNDED,
WORK CLASS CALLSTATEMENTS WORK TYPE CALL ROUTINES IN SCHEMA MYSCHEMA)
```

2. データベース・レベルで MYWORKCLASSSET ワーク・クラス・セットのワーク・クラスに適用される 3 つのワーク・アクションが入った DATABASEACTIONS ワーク・アクション・セットを作成します。

```
CREATE WORK ACTION SET DATABASEACTIONS FOR DATABASE
USING WORK CLASS SET LARGEQUERIES
(WORK ACTION ONECONCURRENTQUERY ON WORK CLASS LARGEESTIMATEDCOST
WHEN CONCURRENTDBCOORDACTIVITIES > 1 AND QUEUEDACTIVITIES > 1 STOP EXECUTION,
WORK ACTION TWOCONCURRENTQUERIES ON WORK CLASS LARGECARDINALITY
WHEN CONCURRENTDBCOORDACTIVITIES > 2 AND QUEUEDACTIVITIES > 3 STOP EXECUTION,
WORK ACTION FIVECALLS ON WORK CLASS CALLSTATEMENTS
WHEN CONCURRENTDBCOORDACTIVITIES > 5 COLLECT ACTIVITY DATA CONTINUE)
```

さらに、いくつかの大規模な管理アプリケーションがデータベースに対して毎日実行されており、これらのアプリケーションを 1 つのリソース・プールで実行することを希望するとします。この目標を達成するために、これらのアプリケーションのために ADMINAPPS というサービス・スーパークラスを作成することができます。アプリケーションごとに、それを ADMINAPPS サービス・スーパークラスにマップするためのワークロードを作成します。

照会 (SELECT ステートメント) が迅速に実行されることは重要であるため、ADMINAPPS サービス・スーパークラスに、これらの照会用の SELECTS というサービス・サブクラスを作成することにします。

SELECT および XQuery ステートメントを SELECTS サービス・サブクラスにマップするには、以下のようにします。

1. データベースを更新しないすべての SELECT ステートメント用のワーク・クラスが入った SELECTDML ワーク・クラス・セットを作成します。

```
CREATE WORK CLASS SET SELECTDML (WORK CLASS SELECTCLASS WORK TYPE READ)
```

2. ADMINAPPSACTIONS ワーク・アクション・セットを作成します。このワーク・アクション・セットには、サービス・スーパークラス・レベルでワーク・アクション・セット SELECTDML のワーク・クラスに適用されるワーク・アクションが入ります。

```
CREATE WORK ACTION SET ADMINAPPSACTIONS FOR SERVICE CLASS ADMINAPPS
USING WORK CLASS SET SELECTDML
(WORK ACTION MAPSELECTS ON WORK CLASS SELECTCLASS MAP ACTIVITY TO SELECTS)
```

例: ALL キーワードを使用して定義されたワーク・クラスの処理

以下の例では、ALL として定義されるワーク・クラスを処理する方法について示します。これは、データベース内で認識されるすべてのアクティビティーを網羅する可能性があるワーク・クラスです。

ALL のタイプのワーク・クラスをマッピング・ワーク・アクションとともに使用した場合、認識されるすべてのデータベース・アクティビティーは、ワーク・アクションで指定されたサービス・サブクラスにマップされます。 ALL の作業タイプのワーク・クラスをしきい値のワーク・アクションとともに使用した場合、しきい値

のタイプによって、しきい値が適用されるデータベース・アクティビティーが決定されます。次のような例について考慮します。

以下のワーク・クラスを使用して Example というワーク・クラス・セットを作成するとします。ワーク・クラスの評価順序は、以下のとおりです。

1. SMALLDML (見積コストが 1000 timeron より小さいすべての DML タイプ SQL 用)。
2. LOADUTIL (ロード・ユーティリティー用)。
3. DDLACTIVITY (すべての DDL アクティビティー用)。
4. ALLACTIVITY (すべてのデータベース・アクティビティー用)。

ALLACTIVITY は最後に評価されるワーク・クラスであり、最初の 3 つのワーク・クラスに対応していないデータベース・アクティビティーを網羅します。

このワーク・クラス・セットの作成用の DDL は次のとおりです。

```
CREATE WORK CLASS SET EXAMPLE
(WORK CLASS SMALLDML WORK TYPE DML FOR TIMERONCOST FROM 0 TO 999,
WORK CLASS LOADUTIL WORK TYPE LOAD, WORK CLASS DDLACTIVITY WORK TYPE DDL,
WORK CLASS ALLACTIVITY WORK TYPE ALL)
```

EXAMPLESERVICECLASS というサービス・スーパークラスがあり、これに SMALLACTIVITY および OTHERACTIVITY という 2 つのサービス・サブクラスが含まれているとします。すべての小さなデータベース・アクティビティーが SMALLACTIVITY サービス・サブクラスで実行されるように、さらに (ロード・ユーティリティーを除く) 認識される他のすべてのデータベース・アクティビティーが OTHERACTIVITY サービス・サブクラスで実行されるようにシステムをセットアップしたいとします。ロード・ユーティリティーをその他のサービス・サブクラスには再マップせずに、代わりにデフォルトのサービス・サブクラスで実行したいものとします。

これらの目標を達成するために、EXAMPLESERVICECLASS サービス・スーパークラスに対してワーク・アクション・セット SERVICECLASSACTIONS をセットアップします。 SERVICECLASSACTIONS ワーク・アクション・セットには、以下のワーク・アクションが含まれます。

表 37. SERVICECLASSACTIONS ワーク・アクション・セット

ワーク・アクション	適用されるワーク・クラス	アクション
MAPDML	SMALLDML	SMALLACTIVITY サービス・サブクラスにマップします
MAPOTHER	ALLACTIVITY	OTHERACTIVITY サービス・サブクラスにマップします
COUNTLOAD	LOADUTIL	LOAD アクティビティーの数をカウントします

このワーク・アクション・セットを作成するための DDL は次のとおりです。

```
CREATE WORK ACTION SET SERVICECLASSACTIONS FOR SERVICE CLASS EXAMPLESERVICECLASS
USING WORK CLASS SET EXAMPLE
(WORK ACTION MAPDML ON WORK CLASS SMALLDML MAP ACTIVITY TO SMALLACTIVITY,
WORK ACTION MAPOTHER ON WORK CLASS ALLACTIVITY MAP ACTIVITY TO OTHERACTIVITY,
WORK ACTION COUNTLOAD ON WORK CLASS LOADUTIL COUNT ACTIVITY)
```

この構成を使用すると、すべての小さな DML は SMALLACTIVITY サービス・サブクラスの下で実行されます。 COUNTLOAD ワーク・アクションは、デフォルトのサービス・サブクラスの下で実行される LOADUTIL ワーク・クラスに適用されます。認識される他のすべてのデータベース・アクティビティーは、OTHERACTIVITY サービス・サブクラスの下で実行されます。LOADUTIL ワーク・クラスに適用されるワーク・アクションがなかったとすれば、LOAD アクティビティーは ALLACTIVITY ワーク・クラスに該当し、MAPOTHER ワーク・アクションが適用されることになります (ワーク・クラスにワーク・アクションが適用されない場合、そのワーク・クラスの下で分類されるアクティビティーはありません)。

注: ALLACTIVITY ワーク・クラスが評価順序の先頭にあったとすれば、認識されるすべてのアクティビティーは OTHERACTIVITY サービス・サブクラスにマップされることになります。

ここで、データベースに対してワーク・アクション・セットを定義し、システム上で並行して実行可能なものを制御するしきい値を適用するとします。以下のワーク・アクションが含まれる DATABASEACTIONS というワーク・アクション・セットを作成することができます。このワーク・アクション・セットを作成するための DML は次のとおりです。

```
CREATE WORK ACTION SET DATABASEACTIONS FOR DATABASE USING WORK CLASS SET EXAMPLE
(WORK ACTION CONCURRENTSMALLDML ON WORK CLASS SMALLDML
WHEN CONCURRENTDBCOORDACTIVITIES > 1000 AND QUEUEDACTIVITIES > 10000
COLLECT ACTIVITY DATA STOP EXECUTION,
WORK ACTION CONCURRENTLOAD ON WORK CLASS LOADUTIL
WHEN CONCURRENTDBCOORDACTIVITIES > 2 AND QUEUEDACTIVITIES > 10
COLLECT ACTIVITY DATA STOP EXECUTION,
WORK ACTION CONCURRENTOTHER ON WORK CLASS ALLACTIVITY
WHEN CONCURRENTDBCOORDACTIVITIES > 100 AND QUEUEDACTIVITIES > 100
COLLECT ACTIVITY DATA STOP EXECUTION,
WORK ACTION MAXCOSTALLOWED ON WORK CLASS ALLACTIVITY
WHEN ESTIMATEDSQLCOST > 1000000 COLLECT ACTIVITY DATA STOP EXECUTION)
```

表 38. DATABASEACTIONS ワーク・アクション・セット

ワーク・アクション	適用されるワーク・クラス	しきい値タイプおよび値	アクション
CONCURRENTSMALLDML	SMALLDML	並行性が 1000 ステートメントまで。キューが 10,000 ステートメントまで。	• 実行の停止 • アクティビティー・データの収集
CONCURRENTLOAD	LOADUTIL	並行性が 2 オカレンスまで。キューが 10 オカレンスまで。	• 実行の停止 • アクティビティー・データの収集
CONCURRENTOTHER	ALLACTIVITY	並行性が 100 アクティビティーまで。キューが 100 アクティビティーまで。	• 実行の停止 • アクティビティー・データの収集

表 38. *DATABASEACTIONS* ワーク・アクション・セット *(続き)*

ワーク・アクション	適用されるワーク・クラス	しきい値タイプおよび値	アクション
MAXCOSTALLOWED	ALLACTIVITY	見積 SQL コストが 1,000,000 timeron まで。	• 実行の停止 • アクティビティー・データの収集

これらのワーク・アクションが適用されると、最大で 1000 個の小さい DML タイプの SQL ステートメント (SMALLDML ワーク・クラスのため) を一度に実行することができます。さらに、最大で 10,000 個のステートメントをキューに入れることができます。一度に実行できるロード・ユーティリティーのオカレンスは 2 つのみであり、最大 10 のオカレンスをキューに入れることができます。 LOAD でなく、小さい DML でもないアクティビティーは一度に 100 のみ実行できます。また、これらのアクティビティーのうち一度にキューに入れることができるのは 100 のみです。いずれの状態においても、キューのしきい値に違反している場合、データベース・アクティビティーを実行することはできず、エラー・メッセージが返されます。

さらに、MAXCOSTALLOWED ワーク・アクションが ALLACTIVITY クラスに適用されます。これは、1,000,000 timeron より大きい見積コストを持つデータベース・アクティビティー (つまり、DML および XQueries ステートメント) は実行できないことを意味します。 MAXCOSTALLOWED ワーク・アクションは ALLACTIVITY ワーク・クラスに適用されますが、このワーク・アクションは見積コストが 1,000,000 timeron より大きいデータベース・アクティビティーにのみ影響を与えます。このワーク・アクションは、見積コストを持たないアクティビティー (DDL など) には影響しません。

例: ワーク・アクション・セットおよびデータベースしきい値の使用

以下の例では、DB2 アクティビティーによって消費されるリソースを制御する、ワーク・アクション・セットおよびしきい値のさまざまな使用方法を示します。ワークロード管理オブジェクトを作成する前に、それらの使用方法について理解しておく必要があります。

ALLSQL というワーク・クラス・セットがあり、それに以下のワーク・クラスが以下に示す順序で含まれているとします。

1. SMALLDML (見積コストが 1,000 timeron より小さいすべての DML タイプ SQL ステートメント用)
2. MEDDML (見積コストが 1,000 timeron から 20,000 timeron までの、すべての DML タイプ SQL ステートメント用)
3. LARGEDML (見積コストが 20,000 timeron より大きいすべての DML タイプ SQL ステートメント用)
4. ALLDDL (すべての DDL タイプ SQL ステートメント用)
5. ALLACTIVITY (すべてのデータベース・アクティビティー用)

これらのワーク・クラスには既に、COUNT ACTIVITY、COLLECT、およびしきい値 (ACTIVITYTOTALTIME しきい値ではない) などのワーク・アクションが適用されています。

大きな DML アクティビティーが 5 時間を超えて実行することを禁止したいものとします。他のすべての SQL は 30 分を超えて実行することができません。以下の 2 つの例は、この目標を達成するために取り得る方法を示しています。

方法 1

1 つの方法は、各ワーク・クラスに ACTIVITYTOTALTIME しきい値を指定してワーク・アクションを以下のようにセットアップすることです。

表 39. 各ワーク・クラスに指定された ACTIVITYTOTALTIME しきい値

ワーク・アクション	適用されるワーク・クラス	しきい値タイプおよび値	アクション
SMALLDMLTIMEALLOWED	SMALLDML	ACTIVITYTOTALTIME < 31 MINUTES	• 実行の停止 • アクティビティー・データの収集
MEDDMLTIMEALLOWED	MEDDML	ACTIVITYTOTALTIME < 31 MINUTES	• 実行の停止 • アクティビティー・データの収集
LARGEDMLTIMEALLOWED	LARGEDML	ACTIVITYTOTALTIME < 5 HOURS	• 実行の停止 • アクティビティー・データの収集
ALLDDLTIMEALLOWED	ALLDDL	ACTIVITYTOTALTIME < 31 minutes	• 実行の停止 • アクティビティー・データの収集
ALLACTIVITYTIMEALLOWED	ALLACTIVITY	ACTIVITYTOTALTIME < 31 minutes	• 実行の停止 • アクティビティー・データの収集

方法 2

もう 1 つの方法として考えられるのは、LARGEDML というワーク・クラスを 1 つだけ使用し、次いでこのワーク・クラスに適用される LARGEDMLTIMEALLOWED という 1 つのワーク・アクションを持ったワーク・アクション・セットをデータベースのために作成するというものです。

表 40. LARGEDML ワーク・クラスに適用される LARGEDMLTIMEALLOWED ワーク・アクション

ワーク・アクション	適用されるワーク・クラス	しきい値タイプおよび値	アクション
LARGEDMLTIMEALLOWED	LARGEDML	ACTIVITYTOTALTIME < 5 HOURS	• 実行の停止 • アクティビティー・データの収集

次に、31 MINUTES 未満という ACTIVITYTOTALTIME しきい値をデータベースに適用します。この方法を使用した場合、LARGEDML ワーク・クラスに対応するアクティビティーにだけ 5 時間のしきい値が適用されます。他のアクティビティー

は分類されずに、31 分未満という ACTIVITYTOTALTIME データベース時間しきい値が適用されることになります。

例: ワーク・アクション・セットを使用して実行中の作業のタイプを判別する

ワーク・クラス・セット、ワーク・クラス、ワーク・アクション・セット、ワーク・アクション、およびワークロード管理モニター・フィーチャーの一部を使用して、システム上で実行されているさまざまな種類の作業、およびその作業の分布を判別することができます。

この作業を行うには、まず対象となるさまざまな作業タイプのためのワーク・クラスの入った、ワーク・クラス・セットを作成します。例えば、システムで実行中の READ アクティビティー、WRITE アクティビティー、DDL アクティビティー、および LOAD アクティビティーがいくつあるかを知りたい場合、以下の例にあるようにワーク・クラス・セット、ACTIVITYTYPES を作成します。

```
CREATE WORK CLASS SET ACTIVITYTYPES
(WORK CLASS READWC WORK TYPE READ,
WORK CLASS WRITEWC WORK TYPE WRITE,
WORK CLASS DDLWC WORK TYPE DDL,
WORK CLASS LOADWC WORK TYPE LOAD)
```

次に、データベース・レベルのワーク・アクション・セット COUNTACTIONS を作成して、ACTIVITYTYPES ワーク・クラス・セットに適用します。そのワーク・アクション・セットには、以下の例にあるように、ACTIVITYTYPES ワーク・クラス・セットにあるそれぞれのワーク・クラスに対する COUNT ACTIVITY ワーク・アクションが含まれます。

```
CREATE WORK ACTION SET COUNTACTIONS FOR DATABASE USING WORK CLASS SET ACTIVITYTYPES
(WORK ACTION COUNTREAD ON WORK CLASSREADWC COUNT ACTIVITY,
WORK ACTION COUNTWRITE ON WORK CLASS WRITEWC COUNT ACTIVITY,
WORK ACTION COUNTDDL ON WORK CLASS DDLWC COUNT ACTIVITY,
WORK ACTION COUNTLOAD ON WORK CLASS LOADWC COUNT ACTIVITY)
```

十分な時間が経過した後、WLM_GET_WORK_ACTION_SET_STATS 表関数を使用して実行されているタイプごとのアクティビティーの数を判別することができます。

```
SELECT SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
LAST_RESET,
SUBSTR(WORK_CLASS_NAME,1,15) AS WORK_CLASS_NAME,
SUBSTR(CHAR(ACT_TOTAL),1,14) AS TOTAL_ACTS
FROM TABLE(WLM_GET_WORK_ACTION_SET_STATS(CAST(NULL AS VARCHAR(128)), -2))
AS WASSTATS WHERE WORK_ACTION_SET_NAME = 'COUNTACTIONS'
ORDER BY WORK_CLASS_NAME, PART
```

例: ワークロード管理の表関数を使用して、異なるレベルで現行システムの動作をモニターする

ワークロード管理機能は、ワークロード管理構成に関するデータを取得するために使用可能なさまざまな表関数を備えています。

DB2 バージョン 9.5 をインストールすると、デフォルトのワークロードおよびサービス・クラスのセットが作成されます。独自のワークロード管理ソリューションを

実装する方法を決定する前に、表関数を使用して、システムで実行されている作業の監視を、デフォルトのワークロード・オカレンス、サービス・クラス、およびアクティビティーの観点から行うことができます。

最初に、サービス・クラス内のワークロード・オカレンスのリストを取得することができます。これを行うには、
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数を使用します。以下の例では、*service_superclass_name* および *service_subclass_name* には空ストリング、および *dbpartitionnum* には -2 (ワイルドカード文字) を渡します。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       SUBSTR(CHAR(COORD_PARTITION_NUM),1,4) AS COORDPART,
       SUBSTR(CHAR(APPLICATION_HANDLE),1,7) AS APPHNDL,
       SUBSTR(CHAR(WORKLOAD_NAME),1,22) AS WORKLOAD_NAME,
       SUBSTR(CHAR(WORKLOAD_OCCURRENCE_ID),1,6) AS WLO_ID
FROM TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES('', '', -2)) AS SCINFO
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME, PART, APPHNDL, WORKLOAD_NAME, WLO_ID
```

システムに 4 つのデータベース・パーティションがあり、照会を発行したときに 2 つのアプリケーションがデータベース上でアクティビティーを実行しているとします。結果は以下のようになります。

```
SUPERCLASS_NAME     SUBCLASS_NAME      PART COORDPART APPHNDL WORKLOAD_NAME          WLO_ID
------------------- ------------------ ---- --------- ------- ----------------------------
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 0    0         1       SYSDEFAULTUSERWORKLOAD 1
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 0    0         2       SYSDEFAULTUSERWORKLOAD 2
```

結果は、両方のワークロード・オカレンスが SYSDEFAULTUSERWORKLOAD ワークロードに割り当てられたことを示しています。さらに、この結果は、両方のワークロード・オカレンスが SYSDEFAULTUSERCLASS サービス・スーパークラスの SYSDEFAULTSUBCLASS サービス・サブクラスに割り当てられており、どちらのワークロード・オカレンスも同じコーディネーター・パーティション (パーティション 0) のものであることも示しています。

次に、WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数を再度使用して、以下のように 2 つのワークロード・オカレンスの接続属性を判別することもできます。

```
SELECT SUBSTR(CHAR(APPLICATION_HANDLE),1,7) AS APPHNDL,
       SUBSTR(CHAR(WORKLOAD_NAME),1,22) AS WORKLOAD_NAME,
       SUBSTR(CHAR(WORKLOAD_OCCURRENCE_ID),1,6) AS WLO_ID,
       SUBSTR(CHAR(SYSTEM_AUTH_ID),1,9) AS SYSAUTHID,
       SUBSTR(CHAR(APPLICATION_NAME),1,15) AS APPLNAME
FROM TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES('', '', 0)) AS SCINFO
ORDER BY APPHNDL, WORKLOAD_NAME, WLO_ID

APPHNDL WORKLOAD_NAME          WLO_ID SYSAUTHID APPLNAME
------- ---------------------- ------ --------- ---------------
1       SYSDEFAULTUSERWORKLOAD 1      LYNN      accountspay
2       SYSDEFAULTUSERWORKLOAD 2      KATE      businessobjects
```

次に、WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数を使用して、いずれかのワークロード・オカレンスの現在のアクティビティーを表示することができます。

```
SELECT SUBSTR(CHAR(COORD_PARTITION_NUM),1,5) AS COORD,
       SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       SUBSTR(CHAR(UOW_ID),1,5) AS UOWID,
```

```
       SUBSTR(CHAR(ACTIVITY_ID),1,5) AS ACTID,
       SUBSTR(CHAR(PARENT_UOW_ID),1,8) AS PARUOWID,
       SUBSTR(CHAR(PARENT_ACTIVITY_ID),1,8) AS PARACTID,
       SUBSTR(ACTIVITY_TYPE,1,9) AS ACTTYPE,
       SUBSTR(CHAR(NESTING_LEVEL),1,7) AS NESTING
FROM TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(1, -2)) AS WLOACTS
ORDER BY PART, UOWID, ACTID

COORD PART UOWID ACTID PARUOWID PARACTID ACTTYPE  NESTING
----- ---- ----- ----- -------- -------- -------- -------
0     0    1     3     -        -        CALL     0
0     0    1     5     1        3        READ_DML 1
0     1    1     5     -        -        READ_DML 1
0     2    1     5     -        -        READ_DML 1
0     3    1     5     -        -        READ_DML 1
```

照会結果は、ワークロード・オカレンス 1 が 2 つのアクティビティーを実行中であることを示しています。 1 つのアクティビティーはストアード・プロシージャー (CALL のアクティビティー・タイプで示される)、そしてもう 1 つのアクティビティーは読み取りを実行する DML アクティビティー (例、SELECT ステートメントなど) です。DML アクティビティーは、ストアード・プロシージャー呼び出しにネストされています。 DML アクティビティーの親作業単位 ID および親アクティビティー ID が CALL アクティビティーの作業単位 ID およびアクティビティー ID と一致しているため、DML アクティビティーがネストされているということがわかります。さらに、DML アクティビティーがデータベース・パーティション 0、1、2、および 3 で実行されていることもわかります。親 ID の情報はコーディネーター・パーティションでのみ取得できます。

現在実行中の個別のアクティビティーに関する詳細情報を取得するには、WLM_GET_ACTIVITY_DETAILS 表関数を使用できます。この表関数は、データベース・パーティションごとに名前と値の対でアクティビティー情報を返します。以下の例では、表関数はデータベース・パーティションごとに 11 のメンバーの名前と値の対のサブセットのみを表示します。名前と値の対の完全なリストについて詳しくは、『WLM_GET_ACTIVITY_DETAILS 表関数』を参照してください。

```
SELECT SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       SUBSTR(NAME, 1, 20) AS NAME,
       SUBSTR(VALUE, 1, 30) AS VALUE
FROM TABLE(WLM_GET_ACTIVITY_DETAILS(1, 1, 5, -2)) AS ACTDETAIL
WHERE NAME IN ('APPLICATION_HANDLE',
               'COORD_PARTITION_NUM',
               'LOCAL_START_TIME',
               'UOW_ID',
               'ACTIVITY_ID',
               'PARENT_UOW_ID',
               'PARENT_ACTIVITY_ID',
               'ACTIVITY_TYPE',
               'NESTING_LEVEL',
               'INVOCATION_ID',
               'ROUTINE_ID')
ORDER BY PART

PART NAME                 VALUE
---- -------------------- ------------------------------
0    APPLICATION_HANDLE   1
0    COORD_PARTITION_NUM  0
0    UOW_ID               1
0    ACTIVITY_ID          5
0    PARENT_UOW_ID        1
0    PARENT_ACTIVITY_ID   3
0    ACTIVITY_TYPE        READ_DML
```

```
0    NESTING_LEVEL        0
0    INVOCATION_ID        1
0    ROUTINE_ID           0
0    LOCAL_START_TIME     2005-11-25-18.52.49.343000
1    APPLICATION_HANDLE   1
1    COORD_PARTITION_NUM  0
1    UOW_ID               1
1    ACTIVITY_ID          5
1    PARENT_UOW_ID        -
1    PARENT_ACTIVITY_ID   -
1    ACTIVITY_TYPE        READ_DML
1    NESTING_LEVEL        0
1    INVOCATION_ID        1
1    ROUTINE_ID           0
1    LOCAL_START_TIME     2005-11-25-18.52.49.598000
```

前述の 3 つの表関数は、システムで実行中の作業に関する概要的な記述を提供します。これらの表関数が作業の状況について提供する情報は、EXECUTING などのアクティビティー状態に制限されます。ある時点で、サービス・クラスで何が起きているかを正確に知るためにさらに調査する場合は、
WLM_GET_SERVICE_CLASS_AGENTS 表関数を実行することができます。

以下の例では、*application_handle* に 1、*dbpartitionnum* に -2 (ワイルドカード文字) を渡すことによって、WLM_GET_SERVICE_CLASS_AGENTS を呼び出します。

```
SELECT SUBSTR(CHAR(APPLICATION_HANDLE),1,7) AS APPHANDLE,
       SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       SUBSTR(CHAR(AGENT_TID),1,9) AS AGENT_TID,
       SUBSTR(AGENT_TYPE,1,11) AS AGENTTYPE,
       SUBSTR(AGENT_STATE,1,10) AS AGENTSTATE,
       SUBSTR(REQUEST_TYPE,1,14) AS REQTYPE,
       SUBSTR(CHAR(UOW_ID),1,6) AS UOW_ID,
       SUBSTR(CHAR(ACTIVITY_ID),1,6) AS ACT_ID
FROM TABLE(WLM_GET_SERVICE_CLASS_AGENTS('', '', 1, -2)) AS SCDETAILS
ORDER BY APPHANDLE, PART, AGENT_TID

APPHANDLE PART AGENT_TID AGENTTYPE   AGENTSTATE REQTYPE      UOW_ID ACT_ID
--------- ---- --------- ----------- ---------- ------------------------
1         0    3         COORDINATOR ACTIVE     FETCH             1      5
1         0    4         PDBSUBAGENT ACTIVE     SUBSECTION:1      1      5
1         1    2         PDBSUBAGENT ACTIVE     SUBSECTION:2      1      5
```

結果は、データベース・パーティション 0 のコーディネーター・エージェントおよびサブエージェント、およびデータベース・パーティション 1 のサブエージェントが、作業単位 ID が 1 でアクティビティー ID が 5 のアクティビティーのために作動していることを示しています。コーディネーター・エージェント情報は、要求がフェッチ要求であることを示しています。

例: サービス・クラスからの特定時点の統計の取得

すべてのアクティビティーは、実行される前にサービス・クラスにマップされます。サービス・クラス統計の表関数を使用し、すべてのデータベース・パーティション上のすべてのサービス・クラスを照会して、特定時点の統計を取得することにより、システムをモニターできます。

以下のステートメントを使用して、アクティビティーの平均存続期間などのサービス・クラス統計を取得することができます。
WLM_GET_SERVICE_SUBCLASS_STATS 表関数の引数に対して空ストリングを渡

した場合、結果はその引数によって制限されることはありません。最後の引数 *dbpartitionnum* の値は -2 (ワイルドカード文字) であり、これはすべてのデータベース・パーティションからのデータが返されることを意味します。

注: 存続時間情報が返されるのは、COLLECT AGGREGATE ACTIVITY DATA で定義されているサービス・クラスについてのみです。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       CAST(COORD_ACT_LIFETIME_AVG / 1000 AS DECIMAL(9,3)) AS AVGLIFETIME,
       CAST(COORD_ACT_LIFETIME_STDDEV / 1000 AS DECIMAL(9,3)) AS STDDEVLIFETIME,
       SUBSTR(CAST(LAST_RESET AS VARCHAR(30)),1,16) AS LAST_RESET
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS('', '', -2)) AS SCSTATS
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME, PART
```

```
SUPERCLASS_NAME     SUBCLASS_NAME      PART AVGLIFETIME STDDEVLIFETIME LAST_RESET
------------------- ------------------ ---- ----------- -------------- ----------------
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 0        691.242         34.322 2006-07-24-11.44
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 1        644.740         22.124 2006-07-24-11.44
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 2        612.431         43.347 2006-07-24-11.44
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 3        593.451         28.329 2006-07-24-11.44
```

さらに、WLM_GET_SERVICE_SUBCLASS_STATS 表関数を使用して、データベース・パーティションごとにサービス・クラスで実行される、コーディネーター・アクティビティーの並行性の最高水準点を取得することもできます。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       CONCURRENT_ACT_TOP AS ACTHIGHWATERMARK
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS('', '',  -2)) AS SCSTATS
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME, PART

SUPERCLASS_NAME     SUBCLASS_NAME      PART ACTHIGHWATERMARK
------------------- ------------------ ---- ----------------
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 0                  10
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 1                   0
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 2                   0
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 3                   0
```

完了したアクティビティーの平均存続期間および数を検討することにより、WLM_GET_SERVICE_SUBCLASS_STATS 表関数の出力を使用して、データベースの各データベース・パーティションのワークロードのロールアップ・ビューを取得することができます。表関数によって返された最高水準点と平均に大きなばらつきがある場合、システム上でワークロードが変更されている可能性があります。

例: ワークロード管理の表関数を使用したデータの集約

ワークロード管理構成では、表データでさまざまな集約を実行して、システムをモニターしたり、起こりうる問題を識別したりすることができます。

以下は、問題を識別するために実行可能なデータ集約の例です。

照会がキューで費やす時間が多すぎるために生じる照会の平均存続期間の増加を識別する

システム全体に渡って各サービス・クラスのコーディネーター・アクティビティーがキューに存在する平均時間を表示することにより、照会がキューで費やす時間が多すぎるために照会の平均存続時間が増加する状態を識別することができます。

以下の例は、各サービス・クラスでコーディネーター・アクティビティーのために平均的な照会がキューで費やす時間 (全データベース・パーティションに渡って合計) の割合を示しています。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       CASE WHEN (SUM(COORD_ACT_COMPLETED_TOTAL) = 0) THEN
         0       ELSE
         SUM(COORD_ACT_QUEUE_TIME_AVG * COORD_ACT_COMPLETED_TOTAL) /
         SUM(COORD_ACT_COMPLETED_TOTAL)
       END AS AVG_QUEUE_TIME,
       CASE WHEN (SUM(COORD_ACT_COMPLETED_TOTAL) = 0) THEN
         0       ELSE
         SUM(COORD_ACT_LIFETIME_AVG * COORD_ACT_COMPLETED_TOTAL) /
         SUM(COORD_ACT_COMPLETED_TOTAL)
       END AS AVG_LIFE_TIME,
       CASE WHEN (SUM(COORD_ACT_COMPLETED_TOTAL) = 0) THEN
         0       ELSE CASE WHEN
         (CAST(SUM(COORD_ACT_LIFETIME_AVG * COORD_ACT_COMPLETED_TOTAL) /
          SUM(COORD_ACT_COMPLETED_TOTAL) AS INTEGER) = 0) THEN
           0         ELSE
           100 * (SUM(COORD_ACT_QUEUE_TIME_AVG * COORD_ACT_COMPLETED_TOTAL) /
                      SUM(COORD_ACT_COMPLETED_TOTAL)) /
                     (SUM(COORD_ACT_LIFETIME_AVG * COORD_ACT_COMPLETED_TOTAL) /
                      SUM(COORD_ACT_COMPLETED_TOTAL))
         END       END AS PERCENT_TIME_QUEUED
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS('', '', -2)) AS STATS
GROUP BY SERVICE_SUPERCLASS_NAME, SERVICE_SUBCLASS_NAME
ORDER BY SERVICE_SUPERCLASS_NAME, SERVICE_SUBCLASS_NAME
```

```
SUPERCLASS_NAME     SUBCLASS_NAME      AVG_QUEUE_TIME           AVG_LIFE_TIME            PERCENT_TIME_QUEUED
------------------- ------------------ ------------------------ ------------------------ ------------------------
SYSDEFAULTMAINTENAN SYSDEFAULTSUBCLASS +0.00000000000000E+000   +0.00000000000000E+000   +0.00000000000000E+000
SYSDEFAULTSYSTEMCLA SYSDEFAULTSUBCLASS +0.00000000000000E+000   +0.00000000000000E+000   +0.00000000000000E+000
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS +2.32860100000000E+005   +8.23421424000000E+005   +2.82800000000000E-001
```

結果は、平均的なアクティビティーがキュー内で費やす時間の割合が約 28% であることを示しています。システムおよびワークロードでの以前の経験から見て、この値が高すぎるか低すぎる場合、しきい値を調整すると、キューイングに費やされる時間の割合に影響する可能性があります。

ワークロードで実行されている照会数の突然の増加を識別する

WL1 というワークロードがあると想定します。システム全体を通じて、ワークロードに対して実行中のネストなしコーディネーター・アクティビティーの合計数を表示することによって、ワークロードで非常に多くの照会が実行されている状態を識別することができます。

```
SELECT SUBSTR(WORKLOAD_NAME,1,22) AS WLNAME,
COUNT(*) AS TOTAL_EXE_ACT
FROM TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES('', '', -2)) AS APPS,
TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(APPS.APPLICATION_HANDLE, -2)) AS APPACTS
WHERE WORKLOAD_NAME = 'WL1' AND
APPS.DBPARTITIONNUM = APPS.COORD_PARTITION_NUM AND
ACTIVITY_STATE = 'EXECUTING' AND
NESTING_LEVEL = 0
```

```
GROUP BY WORKLOAD_NAME

WLNAME               TOTAL_EXE_ACT
-------------------- -------------
WL1                  5
```

例: ワークロード管理構成におけるヒストグラムからの平均および標準偏差の計算

ヒストグラムの使用方法の 1 つは、アクティビティーの存続期間の標準偏差を入手することです。このトピックの例では、この統計の計算でビンを使用する方法を示します。

アクティビティーごとの平均存続期間の計算は、有用な情報の 1 つです。ただし、平均だけではユーザーの体感について正確に表すことはできません。アクティビティーの存続期間の変動性が大きいと、サポートを受けているユーザーが、照会の実行が高速なとき (望ましい状態) と、低速なとき (許容されないことのある状態) を体感されるかもしれません。アクティビティーの存続期間の目標を定義する場合、アクティビティーの平均存続期間だけでなく、アクティビティーの存続時間の標準偏差も重要になります。ユーザーが実際に体感しているのが実測上の平均であるようにするためには、変動性について理解し、それを制御する必要があります。

ワークロード管理構成において、統計はデータベース・パーティションごとに収集されます。以下の例は、単一データベース・パーティションのアクティビティーの平均存続期間を取得する方法を示しています。

単一パーティション環境において、以下のビンが含まれるヒストグラムがあるとします。実際のヒストグラムではさらに多くのビンが存在しますが、ここでは例を簡潔にするためにビンの数は 8 に制限されています。

```
ビン 1 - 0 から 2 秒
ビン 2 - 2 から 4 秒
ビン 3 - 4 から 8 秒
ビン 4 - 8 から 16 秒
ビン 5 - 16 から 32 秒
ビン 6 - 32 から 64 秒
ビン 7 - 64 から 128 秒
ビン 8 - 128 秒から無限大
```

x から y の範囲のビンに該当する照会の平均応答時間を (x + y)/2 とすることで、平均の近似値を計算できます。次に、この数値とこのビンに入った照会数とを乗算し、すべてのビンに渡って合計してから、その和を全体のカウントで除算することができます。前述の例の場合に、各ビンの平均応答時間を以下のようであるとします。

```
ビン 1 の平均存続期間 = (0+2)/2 = 1
ビン 2 の平均存続期間 = (2+4)/2 = 3
ビン 3 の平均存続期間 = (4+8)/2 = 6
ビン 4 の平均存続期間 = (8+16)/2 = 12
ビン 5 の平均存続期間 = (16+32)/2 = 24
ビン 6 の平均存続期間 = (32+64)/2 = 48
ビン 7 の平均存続期間 = (64+128)/2 = 96
```

以下のヒストグラムが測定期間中に収集されたと仮定します。

ビン 1 カウント	ビン 2 カウント	ビン 3 カウント	ビン 4 カウント	ビン 5 カウント	ビン 6 カウント	ビン 7 カウント	ビン 8 カウント
20	30	80	10	5	3	2	0

平均存続期間を計算するには、ビン 8 は空でなければなりません。ビン 8 は、範囲の上限を変更する必要がある場合に通知するという目的でのみ存在します。このため、範囲の上限を指定する必要があります。

データベース・パーティション 1 の平均存続期間を以下のように概算することができます。

```
平均存続時間 = (20 x 1 + 30 x 3 + 80 x 6 + 10 x 12 + 5 x 24 + 3 x 48 + 2 x 96) / 150
              = (20 + 90 + 480 + 120 + 120 + 144 + 192) / 150
              = 1166 / 150
              = 7.77 秒
```

存続期間の標準偏差は、以下のように概算することができます。

標準偏差 = $[(20 \times (1 - 7.77)2 + 30 \times (3 - 7.77)2 + \ldots) / 150]^{1/2}$

パーティション・データベース環境の場合、全データベース・パーティションの各ビンのカウントを加算し、全データベース・パーティションに渡る結合ヒストグラムを計算することで、平均および標準偏差を計算することができます。

例えば、データベースに 2 つのパーティションが含まれており、ヒストグラムのビンのサイズは上で説明されているとおりで、ヒストグラムには以下のデータが含まれるとします。

データベース・パーティション	ビン 1 カウント	ビン 2 カウント	ビン 3 カウント	ビン 4 カウント	ビン 5 カウント	ビン 6 カウント	ビン 7 カウント	ビン 8 カウント
1	20	30	80	10	5	3	2	0
2	1	5	20	20	4	0	0	0

ビンのサイズはすべてのデータベース・パーティションで同じであるため、ヒストグラム全体は以下のように簡単に計算することができます。

ビン 1 カウント	ビン 2 カウント	ビン 3 カウント	ビン 4 カウント	ビン 5 カウント	ビン 6 カウント	ビン 7 カウント	ビン 8 カウント
21	35	100	30	9	3	2	0

結合ヒストグラムから、全体の存続期間の平均および標準偏差を、単一パーティション環境で計算したときと同じ方法で以下のように計算できます。

```
平均存続時間 = (21 x 1 + 35 x 3 + 100 x 6 + 30 x 12 + 9 x 24 + 3 x 48 + 2 x 96) / 200
              = (21 + 105 + 600 + 360 + 216 + 144 + 192) / 200
              = 1638 / 200
              = 8.19 秒
```

標準偏差 = $[(21 \times (1 - 8.19)^2 + 35 \times (3 - 7.77)^2 + \ldots) / 200]^{1/2}$

例: サービス・クラス関連のシステム・スローダウンの分析

システム・スローダウン (例、一部のアプリケーションを完了するのに予想以上の時間がかかる、など) に気付き、その問題がサービス・クラスの構成に関連しているかどうかが不確かな場合、表関数のデータを使用して問題を調べたり、必要に応じて訂正したりすることができます。

最初に、サービス・クラスで何が起きているか概要を把握します。この概要には、アクティビティーの平均存続期間、異常終了ではなく正常に完了したアクティビテ

ィーの数、およびシステム内の並行コーディネーター・アクティビティーの最高水準点が含まれるはずです。この情報を取得するには、表関数 WLM_GET_SERVICE_CLASS_STATS から取得したデータを使用して、複数のサービス・クラスとデータベース・パーティションにまたがる集約を伴う、一般照会を作成します。すべてのデータベース・パーティションのすべてのサービス・クラスにおいてデータを収集することを指定するには、最初および 2 番目の引数を空ストリングに設定し、3 番目の引数を -2 (ワイルドカード文字) に設定します。照会は次のようになります。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(SUM(COORD_ACT_COMPLETED_TOTAL)),1,13) AS ACTSCOMPLETED,
       SUBSTR(CHAR(SUM(COORD_ACT_ABORTED_TOTAL)),1,11) AS ACTSABORTED,
       SUBSTR(CHAR(MAX(CONCURRENT_ACT_TOP)),1,6) AS ACTSHW,
       CAST(CASE WHEN SUM(COORD_ACT_COMPLETED_TOTAL) = 0 THEN 0
                 ELSE SUM(COORD_ACT_COMPLETED_TOTAL * COORD_ACT_LIFETIME_AVG)
                 / SUM(COORD_ACT_COMPLETED_TOTAL) END / 1000 AS DECIMAL(9,3))
       AS ACTAVGLIFETIME
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS ('', '', -2)) AS SCSTATS
GROUP BY SERVICE_SUPERCLASS_NAME, SERVICE_SUBCLASS_NAME
ORDER BY SERVICE_SUPERCLASS_NAME, SERVICE_SUBCLASS_NAME

SUPERCLASS_NAME     SUBCLASS_NAME      ACTSCOMPLETED ACTSABORTED ACTSHW ACTAVGLIFETIME
------------------- ------------------ ------------- ----------- ------ --------------
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 8             0           1               3.750
BI_APPS             SYSDEFAULTSUBCLASS 4             0           1              14.230
BATCH               SYSDEFAULTSUBCLASS 1             0           1              25.600
```

以前には、この照会で以下の結果が報告されたとします。

```
SUPERCLASS_NAME     SUBCLASS_NAME      ACTSCOMPLETED ACTSABORTED ACTSHW ACTAVGLIFETIME
------------------- ------------------ ------------- ----------- ------ --------------
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 8             0           1               3.750
BI_APPS             SYSDEFAULTSUBCLASS 4             0           1               5.230
BATCH               SYSDEFAULTSUBCLASS 1             0           1              25.600
```

この照会によって返されるデータは、スローダウンが BI_APPS サービス・クラスで発生していることを示すのに十分かもしれません。そのアクティビティーの平均存続期間が通常よりもかなり高いことを示しているためです。この状態は、その特定のサービス・クラスで使用可能なリソースが使い尽くされようとしていることを示している可能性があります。

すべてのデータベース・パーティションのサービス・クラスの平均では問題を切り分けることができない場合、各データベース・パーティションの平均値を分析することを考慮してください。各データベース・パーティションの平均を全体の平均に集約すると、データベース・パーティション間の大きな不整合が隠されてしまうおそれがあります。この場合、すべてのデータベース・パーティションがコーディネーター・パーティションとして使用されていることが前提になっています。この前提が正しくない場合、コーディネーター・パーティション以外で計算される平均存続期間はゼロになります。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       CAST(COORD_ACT_LIFETIME_AVG / 1000 AS DECIMAL(9,3)) AS AVGLIFETIME
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS('', '', -2)) AS SCSTATS
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME

SUPERCLASS_NAME     SUBCLASS_NAME      PART AVGLIFETIME
------------------- ------------------ ---- -----------
```

```
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 0          3.425
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 1          2.752
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 2          8.230
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 3          0.593
```

この例では、データベース・パーティション 2 は通常よりも多くの作業を受け取っている可能性があります。アクティビティーの平均存続期間が他のデータベース・パーティションの平均存続期間よりも非常に高いためです。

システム・スローダウンが発生する原因としては多くの異なる状態が考えられます。以下の原則を使用して、ワークロード管理の表関数によって提供される情報を最大限に活用してください。

- アプリケーション・ロジックおよび環境のレベル (分離レベルなど) での大量のロック競合を解決します。
- サービス・クラスがそのしきい値レベル (並行要求の数など) に近づいている場合、しきい値を増やす必要があるかもしれません。
- サービス・クラスに割り当てられたリソースが使い尽くされようとしている場合、オペレーティング・システム・サービス・クラスへのマッピングが問題の原因となっている可能性があります (つまり、サービス・クラスに対応するオペレーティング・システムのサービス・クラスが十分な CPU、I/O 帯域幅、および他のリソースを得られていないということです)。
- サービス・クラスで実行されているアクティビティーの数が予想よりも多い可能性があり、これにより、通常よりも多くのリソースが消費されている可能性があります。完了したアクティビティーの数を確認して、サービス・クラスで実行中の作業量が妥当であるかどうかを判別してください。
- 予想よりも多くのアクティビティーがサブミットされており、並行性しきい値が定義されている場合、アクティビティーがキューでより多くの時間を費やしている可能性があります。アクティビティーの平均キュー時間が平均存続期間と同じだけ増加しているかどうかを確認してください。同じ量だけ増加している場合、キューの動作は予想どおりです。しかし、存続時間が容認できないものである場合は、さらに多くのリソースをサービス・クラスに割り振り、並行性しきい値を減らすことを考慮してください。

例: ワークロード関連のシステム・スローダウンの調査

システム・スローダウン (例、一部のアプリケーションを完了するのに予想以上の時間がかかる、など) に気付き、その問題がワークロードの構成に関連しているかどうかが不確かな場合、表関数のデータを使用して問題を調べたり、必要に応じて訂正したりすることができます。

最初に、WLM_GET_SERVICE_SUBCLASS_STATS 表関数からのデータを使用して、複数のサービス・クラスとデータベース・パーティションにまたがるデータを集約する照会を作成します。すべてのデータベース・パーティションのすべてのサービス・クラスにおいてデータを収集することを指定するには、最初および 2 番目の引数を空ストリングに設定し、3 番目の引数を -2 (ワイルドカード文字) に設定します。

照会は次のようになります。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(SUM(COORD_ACT_COMPLETED_TOTAL)),1,13) AS ACTSCOMPLETED,
       SUBSTR(CHAR(SUM(COORD_ACT_ABORTED_TOTAL)),1,11) AS ACTSABORTED,
       SUBSTR(CHAR(MAX(CONCURRENT_ACT_TOP)),1,6) AS ACTSHW,
       CAST(CASE WHEN SUM(COORD_ACT_COMPLETED_TOTAL) = 0 THEN 0
                 ELSE SUM(COORD_ACT_COMPLETED_TOTAL * COORD_ACT_LIFETIME_AVG)
                  / SUM(COORD_ACT_COMPLETED_TOTAL) END / 1000 AS DECIMAL(9,3))
       AS ACTAVGLIFETIME
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS('', '', -2)) AS SCSTATS
GROUP BY SERVICE_SUPERCLASS_NAME, SERVICE_SUBCLASS_NAME
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME
```

```
SUPERCLASS_NAME     SUBCLASS_NAME      ACTSCOMPLETED ACTSABORTED ACTSHW ACTAVGLIFETIME
------------------- ------------------ ------------- ----------- ------ --------------
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 20            0           1               3.750
SUP1                SUB1               40            0           8               7.223
```

先述の例のデータでは、SUP1 サービス・スーパークラスの SUB1 サービス・サブクラスは、通常よりも多くのアクティビティーを同時に実行しています。さらに調査する場合、このサービス・クラスにマップするワークロードの統計を調べることもできます。照会は次のようになります。

```
SELECT SUBSTR(WLSTATS.WORKLOAD_NAME,1,22) AS WL_NAME,
       SUBSTR(CHAR(WLSTATS.DBPARTITIONNUM),1,4) AS PART,
       CONCURRENT_WLO_TOP AS WLO_HIGH_WTRMRK,
       CONCURRENT_WLO_ACT_TOP AS WLO_ACT_HIGH_WTRMRK
FROM TABLE(WLM_GET_WORKLOAD_STATS('', -2)) AS WLSTATS,
     TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES('', '', -2)) AS SCWLOS
WHERE WLSTATS.WORKLOAD_NAME = SCWLOS.WORKLOAD_NAME
AND SCWLOS.SERVICE_SUPERCLASS_NAME = 'SUP1'
AND SCWLOS.SERVICE_SUBCLASS_NAME = 'SUB1'
ORDER BY WL_NAME, PART;
```

```
WL_NAME                PART WLO_HIGH_WTRMRK WLO_ACT_HIGH_WTRMRK
---------------------- ---- --------------- -------------------
LYNNSALES                 0               2                   8
LYNNSALES                 1               0                   0
SYSDEFAULTUSERWORKLOAD    0               1                   1
SYSDEFAULTUSERWORKLOAD    1               0                   0
```

出力は、LYNNSALES ワークロードのアプリケーションが 8 個のアクティビティーを並行してサブミットしたことを示しています。各ワークロード・オカレンスのコーディネーター・アクティビティーの並行性を制限するには、しきい値の追加を考慮してください。

例: アクティビティー・タイプごとのワークロードの分析

実行中のアクティビティーのタイプに応じて、ワークロード管理表関数を使用して環境内のワークロードを調べることができます。

状況によっては、LOAD アクティビティーなどの特定のタイプのアクティビティーの動作を把握したい場合があります。例えば、以下のようにして現在システム内に存在する LOAD アクティビティー数を知ることができます。

```
SELECT COUNT(*)
FROM TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(CAST(NULL AS BIGINT), -2))
AS ACTS
WHERE ACTIVITY_TYPE = 'LOAD'
```

以下の例に示されているように、WLM_GET_WORK_ACTION_SET_STATS 表関数を使用すると、ワークロード管理統計を最後にリセットして以降にサブミットされた特定のタイプのアクティビティー数のカウントを取得できます。タイプ READ のアクティビティーおよびタイプ LOAD のアクティビティー用の READCLASS ワーク・クラスと LOADCLASS ワーク・クラスがあるとします (それぞれのワーク・クラス用にワーク・アクションもなければなりません。ワーク・アクションがないと、アクティビティーはワーク・クラスに分類されません)。* は、READCLASS または LOADCLASS ワーク・クラスに分類されない他のすべてのアクティビティーを表します。

```
SELECT SUBSTR(WORK_ACTION_SET_NAME,1,18) AS WORK_ACTION_SET_NAME,
       SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       SUBSTR(WORK_CLASS_NAME,1,15) AS WORK_CLASS_NAME,
       LAST_RESET,
       SUBSTR(CHAR(ACT_TOTAL),1,14) AS TOTAL_ACTS
FROM TABLE(WLM_GET_WORK_ACTION_SET_STATS('', -2)) AS WASSTATS
ORDER BY WORK_ACTION_SET_NAME, WORK_CLASS_NAME, PART

WORK_ACTION_SET_NAME PART WORK_CLASS_NAME LAST_RESET                TOTAL_ACTS
-------------------- ---- --------------- ------------------------- ----------
AdminActionSet       0    ReadClass       2005-11-25-18.52.49.343000 8
AdminActionSet       1    ReadClass       2005-11-25-18.52.50.478000 0
AdminActionSet       0    LoadClass       2005-11-25-18.52.49.343000 2
AdminActionSet       1    LoadClass       2005-11-25-18.52.50.478000 0
AdminActionSet       0    *               2005-11-25-18.52.50.478000 0
AdminActionSet       1    *               2005-11-25-18.52.50.478000 0
```

LOAD アクティビティーを特定のサービス・サブクラスにマップするワーク・アクション・セットを作成すると、LOAD アクティビティーの平均存続時間を表示できます。例えば、LOAD アクティビティーを、サービス・スーパークラス MYSUPERCLASS の下のサービス・サブクラス LOADSERVICECLASS にマップするとします。その場合、WLM_GET_SERVICE_CLASS_STATS 表関数を次のように照会できます。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       CAST(COORD_ACT_LIFETIME_AVG / 1000 AS DECIMAL(9,3)) AS AVGLIFETIME
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS('MYSUPERCLASS', 'LOADSERVICECLASS', -2))
AS SCSTATS
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME, PART

SUPERCLASS_NAME     SUBCLASS_NAME      PART AVGLIFETIME
------------------- ------------------ ---- ----------------------
SYSDEFAULTUSERCLASS LOADSERVICECLASS   0                  4691.242
SYSDEFAULTUSERCLASS LOADSERVICECLASS   1                  4644.740
SYSDEFAULTUSERCLASS LOADSERVICECLASS   2                  4612.431
SYSDEFAULTUSERCLASS LOADSERVICECLASS   3                  4593.451
```

例: ハング・アクティビティーの識別

ワークロード管理表関数を使用すると、データ・サーバー内の特定のアクティビティーを識別したり、必要に応じてアプリケーション全体を終了させることなくそのアクティビティーを取り消したりするタスクを単純化することができます。

ハング・アクティビティーの識別

以下は、ハング照会の識別の例です。SalesReport アプリケーションを実行中の販売部のユーザーから、アプリケーションがハングしたとの苦情が報告されたとします。

アプリケーション・ハンドルを識別してから、WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数を使用して、このアプリケーションで現在実行中のすべてのアクティビティーを調べます。例えば、アプリケーション・ハンドルが 1 の場合、照会は以下のようになります。

```
SELECT SUBSTR(CHAR(COORD_PARTITION_NUM),1,5) AS COORD,
SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
SUBSTR(CHAR(UOW_ID),1,5) AS UOWID,
SUBSTR(CHAR(ACTIVITY_ID),1,5) AS ACTID,
SUBSTR(CHAR(PARENT_UOW_ID),1,8) AS PARUOWID,
SUBSTR(CHAR(PARENT_ACTIVITY_ID),1,8) AS PARACTID,
SUBSTR(ACTIVITY_TYPE,1,8) AS ACTTYPE,
SUBSTR(CHAR(NESTING_LEVEL),1,7) AS NESTING
FROM TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(1, -2)) AS WLOACTS
ORDER BY PART, UOWID, ACTID

COORD PART UOWID ACTID PARUOWID PARACTID ACTTYPE NESTING
----- ---- ----- ----- -------- -------- -------- -------
    0    0     2     3        -        -     CALL       0
    0    0     2     5        2        3 READ_DML       1
```

該当するアクティビティーは、作業単位 ID が 2、アクティビティー ID が 5 であると識別されます。その後、WLM_GET_SERVICE_CLASS_AGENTS 表関数を以下のように使用して、このアクティビティーでエージェントが行っている事柄を把握できます。

```
SELECT APPLICATION_HANDLE,
UOW_ID,
ACTIVITY_ID,
SUBSTR(REQUEST_TYPE,1,8) AS REQUEST_TYPE,
SUBSTR(EVENT_TYPE,1,8) AS EVENT_TYPE,
SUBSTR(EVENT_OBJECT,1,8) AS EVENT_OBJECT
FROM TABLE(WLM_GET_SERVICE_CLASS_AGENTS('', '',
CAST(NULL AS BIGINT),
-2)) AS AGENTS
WHERE APPLICATION_HANDLE = 1
AND UOW_ID = 2
AND ACTIVITY_ID = 5
```

例えば、このアクティビティーがキューに入れられる場合や、実行中、またはロックで待機中の場合があります。アクティビティーがキューに入れられた場合の結果は、次のようになります。

```
APPLICATION_HANDLE UOW_ID ACTIVITY_ID REQUEST_TYPE EVENT_TYPE EVENT_OBJECT
------------------ ------ ----------- ------------ ---------- ------------
                 1      2           5         OPEN       WAIT    WLM_QUEUE
```

アクティビティーが実行中の場合の結果は、次のようになります。

```
APPLICATION_HANDLE UOW_ID ACTIVITY_ID REQUEST_TYPE EVENT_TYPE EVENT_OBJECT
------------------ ------ ----------- ------------ ---------- ------------
                 1      2           5          OPEN   PROCESS      REQUEST
```

アクティビティーがロック上で待機中の場合の結果は、次のようになります。

```
APPLICATION_HANDLE UOW_ID ACTIVITY_ID REQUEST_TYPE EVENT_TYPE EVENT_OBJECT
------------------ ------ ----------- ------------ ---------- ------------
                 1      2           5         OPEN    ACQUIRE         LOCK
```

アクティビティーが現在行っている事柄を確認したなら、それに応じて以下のように続行できます。

- アクティビティーがキューに存在する場合、照会の実行時間が長すぎてもはやユーザーが結果に関心がない、または照会が消費しているリソース量が多すぎると判断するのであれば、その照会を取り消せます。
- 重要なアクティビティーがキューに存在する場合、現在実行中の作業のうちそれほど重要でない他のものを取り消すことを考慮するか (並行性を減らして重要なアクティビティーがキューに入れられたままにする)、その作業がハングしているわけではなく、実行の機会を待機しているだけであることが分かってユーザーは納得する場合もあります。
- アクティビティーがロックを待機している場合、スナップショット・モニターを使用してアプリケーションが待機しているロックについて調査できます。
- アクティビティーが待機しているロックが優先度の低いアクティビティーによって保持されているのであれば、その優先度の低いアクティビティーを取り消すことを考慮してください。

アクティビティー 5 が実行中の DML ステートメントについて把握することが役立つ場合もあります。アクティブ・アクティビティー・イベント・モニターがあると仮定します。WLM_CAPTURE_ACTIVITY_IN_PROGRESS プロシージャーを実行すると、アクティビティー 5 が実行中の DML ステートメントに関する情報および他の情報をキャプチャーできます。WLM_CAPTURE_ACTIVITY_IN_PROGRESS プロシージャーを使用すると、ステートメント・イベント・モニターとは異なり、その時点で実行されているすべてのステートメントではなく特定の照会に関する情報をキャプチャーできます。また WLM_GET_ACTIVITY_DETAILS を使用すると、ステートメント・テキストを最初の 1024 文字に切り捨てた形で取得できます。

アクティビティーを取り消す必要があると判断する場合、WLM_CANCEL_ACTIVITY ルーチンを以下のように使用すると、アクティビティーを発行したアプリケーションを終了させなくてもそのアクティビティーを取り消せます。

```
CALL WLM_CANCEL_ACTIVITY (1, 2, 5)
```

アクティビティーを発行したアプリケーションは SQL4725N エラーを受け取ります。負の SQL コードを処理するアプリケーションはこの SQL コードを処理できます。

ロック競合によるアクティビティー・ハングの識別

あるユーザーからアプリケーションのハングに関して苦情が報告されたとします。さらに、そのハングしたアプリケーションのアプリケーション名または許可 ID のいずれかが分かっていると想定してください。この情報を用いて、LIST APPLICATIONS コマンドを使用すると、アプリケーション・ハンドルを取得できます。LIST APPLICATIONS コマンドによって戻されたアプリケーション・ハンドル

が 2 であるとすると、WLM_GET_SERVICE_CLASS_AGENTS 表関数を使用してこのアクティビティーを操作しているエージェントを判別できます。照会は次のようになります。

```
SELECT SUBSTR(CHAR(APPLICATION_HANDLE),1,7) AS APPHANDLE,
       SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       SUBSTR(CHAR(AGENT_TID),1,9) AS AGENT_TID,
       SUBSTR(AGENT_TYPE,1,11) AS AGENTTYPE,
       SUBSTR(EVENT_OBJECT,1,11) AS EVENTOBJECT,
       SUBSTR(REQUEST_TYPE,1,7) AS REQTYPE,
       SUBSTR(CHAR(UOW_ID),1,6) AS UOW_ID,
       SUBSTR(CHAR(ACTIVITY_ID),1,6) AS ACT_ID
FROM TABLE(WLM_GET_SERVICE_CLASS_AGENTS('', '', 2, -2)) AS SCDETAILS
ORDER BY APPHANDLE, PART, AGENT_TID

APPHANDLE PART AGENT_TID AGENTTYPE   EVENTOBJECT  REQTYPE UOW_ID ACT_ID
--------- ---- --------- ----------- -----------  ------- ------ ------
2         0    1         COORDINATOR REQUEST      OPEN    2      1
2         1    3         SUBAGENT    LOCK         -       2      1
```

エージェント 1 がリモート応答を待機中であることを結果は示しています。同じアクティビティーを操作しているリモート・パーティション上のエージェントを見ると、そのエージェントがロックを取得するために待機中であることが EVENTOBJECT フィールドから分かります。

次のステップでは、ロックの所有者を判別します。この情報は以下の例に示されているように、モニター・スイッチをオンにして、スナップショット・モニター表関数を使用すると取得できます。

```
SELECT AGENT_ID AS WAITING_FOR_LOCK,
       SUBSTR(APPL_ID_HOLDING_LK,1,40) AS HOLDING_LOCK,
       CAST(LOCK_MODE_REQUESTED AS SMALLINT) AS WANTED,
       CAST(LOCK_MODE AS SMALLINT) AS HELD
FROM TABLE(SNAPSHOT_LOCKWAIT('SAMPLE',-1)) AS SLW

WAITING_FOR_LOCK     HOLDING_LOCK                               WANTED HELD
-------------------- ---------------------------------------- ------ ------
                   2 *LOCAL.DB2.060131021547                       9      5
```

また以下の一連のコマンドを使用しても、ロック所有者を判別できます。

```
db2pd -db database alias -locks
db2pd –db database alias -transactions
```

ハング・アクティビティーを取り消す場合には、WLM_CANCEL_ACTIVITY プロシージャーを使用できます。ハング・アプリケーションを正常終了させる方がロックを所有しているアプリケーションを正常終了させるよりも重要な場合には、ロックを所有しているアプリケーションを強制終了させます。

例: 事後分析のためのアクティビティーに関する情報のキャプチャー

ワークロード管理ツールを使用して、後ほど分析するために、アクティビティーに関する情報をキャプチャーできます。

MYSCHEMA.MYSLOWSTP というストアード・プロシージャーがあり、通常より実行速度が遅いとします。この状況に関して苦情が報告されたので、速度が低下した原因を調査することにします。ストアード・プロシージャーの実行中にその調査を

行うことが実際的ではない場合には、そのストアード・プロシージャー・アクティビティーおよびそのアクティビティーにネストしているすべてのアクティビティーに関する情報をキャプチャーできます。

DB2ACTIVITIES というアクティブ・アクティビティー・イベント・モニターがあるとします。CALL ステートメントのワーク・クラスを作成して、MYSCHEMA.MYSLOWSTP ストアード・プロシージャーのスキーマに適用できます。その後、CALL アクティビティーおよびすべてのネストされたアクティビティーを、アクティビティー・コレクションが使用可能なサービス・クラスにマップするようなワーク・アクションを作成できます。CALL アクティビティーおよびそのすべてのネストされているアクティビティーはイベント・モニターに送信されます。以下は、ワークロード管理オブジェクトを作成するために必要な DDL の例です。

```
CREATE SERVICE CLASS SC1;
CREATE WORKLOAD WL1 APPLNAME ('DB2BP') SERVICE CLASS SC1;
CREATE SERVICE CLASS PROBLEMQUERIESSC UNDER SC1 COLLECT ACTIVITY DATA ON COORDINATOR WITH DETAILS;

CREATE WORK CLASS SET PROBLEMQUERIES
(WORK CLASS CALLSTATEMENTS WORK TYPE CALL ROUTINES IN SCHEMA MYSCHEMA);

CREATE WORK ACTION SET DATABASEACTIONS FOR SERVICE CLASS SC1 USING WORK CLASS SET PROBLEMQUERIES
(WORK ACTION CAPTURECALL ON WORK CLASS CALLSTATEMENTS MAP ACTIVITY WITH NESTED TO PROBLEMQUERIESSC);
```

MYSCHEMA.MYSLOWSTP ストアード・プロシージャーの実行後に以下の照会を発行すると、アクティビティーのアプリケーション・ハンドル、作業単位 ID、およびアクティビティー ID を取得できます。

```
SELECT AGENT_ID,
       UOW_ID,
       ACTIVITY_ID
FROM ACTIVITY_DB2ACTIVITIES
WHERE SC_WORK_ACTION_SET_ID = (SELECT ACTIONSETID
                           FROM SYSCAT.WORKACTIONSETS
                           WHERE ACTIONSETNAME = 'DATABASEACTIONS')
AND SC_WORK_CLASS_ID = (SELECT WORKCLASSID
                     FROM SYSCAT.WORKCLASSES
                     WHERE WORKCLASSNAME = 'CALLSTATEMENTS'
                     AND WORKCLASSSETID =
                     (SELECT WORKCLASSSETID FROM SYSCAT.WORKACTIONSETS WHERE ACTIONSETNAME
                     = 'DATABASEACTIONS'));
```

キャプチャーされたアクティビティーのアプリケーション・ハンドルが 1 で、作業単位 ID は 2、アクティビティー ID は 3 であると想定すると、以下の結果が生成されます。

```
AGENT_ID             UOW_ID      ACTIVITY_ID
==================== =========== ===========
                   1           2           3
```

この情報を使用して、ACTIVITY_DB2ACTIVITIES 表および ACTIVITYSTMT_DB2ACTIVITIES 表に対して以下の照会を発行し、アクティビティーが時間を費やす対象を判別できます。

```
WITH RAH (LEVEL, APPL_ID, PARENT_UOW_ID, PARENT_ACTIVITY_ID,
          UOW_ID, ACTIVITY_ID, STMT_TEXT, TIME_CREATED, TIME_COMPLETED) AS
  (SELECT 1, ROOT.APPL_ID, ROOT.PARENT_UOW_ID,
          ROOT.PARENT_ACTIVITY_ID, ROOT.UOW_ID, ROOT.ACTIVITY_ID,
          ROOTSTMT.STMT_TEXT, ROOT.TIME_CREATED, ROOT.TIME_COMPLETED
   FROM ACTIVITY_DB2ACTIVITIES ROOT, ACTIVITYSTMT_DB2ACTIVITIES ROOTSTMT
   WHERE ROOT.APPL_ID = ROOTSTMT.APPL_ID AND ROOT.AGENT_ID = 1
```

```
    AND ROOT.UOW_ID = ROOTSTMT.UOW_ID AND ROOT.UOW_ID = 2
    AND ROOT.ACTIVITY_ID = ROOTSTMT.ACTIVITY_ID AND ROOT.ACTIVITY_ID = 3
  UNION ALL
   SELECT PARENT.LEVEL +1, CHILD.APPL_ID, CHILD.PARENT_UOW_ID,
          CHILD.PARENT_ACTIVITY_ID, CHILD.UOW_ID,
          CHILD.ACTIVITY_ID, CHILDSTMT.STMT_TEXT, CHILD.TIME_CREATED,
          CHILD.TIME_COMPLETED
   FROM RAH PARENT, ACTIVITY_DB2ACTIVITIES CHILD,
        ACTIVITYSTMT_DB2ACTIVITIES CHILDSTMT
   WHERE PARENT.APPL_ID = CHILD.APPL_ID AND
         CHILD.APPL_ID = CHILDSTMT.APPL_ID AND
         PARENT.UOW_ID = CHILD.PARENT_UOW_ID AND
         CHILD.UOW_ID = CHILDSTMT.UOW_ID AND
         PARENT.ACTIVITY_ID = CHILD.PARENT_ACTIVITY_ID AND
         CHILD.ACTIVITY_ID = CHILDSTMT.ACTIVITY_ID AND
         PARENT.LEVEL < 64
  )
SELECT UOW_ID, ACTIVITY_ID, SUBSTR(STMT_TEXT,1,40),
               TIMESTAMPDIFF(2, CHAR(TIME_COMPLETED - TIME_CREATED)) AS
               LIFE_TIME
FROM RAH
  ORDER BY UOW_ID, ACTIVITY_ID;
```

結果は以下のようになります。

```
UOW_ID ACTIVITY_ID STMT_TEXT                              LIFE_TIME
====== =========== ==================================== =============
2      3           CALL SLOWPROC                                 1000
2      4           SELECT COUNT(*) FROM ORG                         1
2      5           SELECT * FROM MYHUGETABLE                      999
```

ストアード・プロシージャーがその時間のほとんどを費やしたのは MYHUGETABLE 表の照会であることを、この結果は示しています。次のステップは、MYHUGETABLE 表に対して加えられたどの変更が、その表に対する照会の実行速度を低下させる原因となっているかを調査することです。

多くのストアード・プロシージャーが同時に実行されると、分析の実行時にオーバーヘッドが一層大きくなります。この問題を解決するには、ワークロードおよびサービス・クラスを作成して、特定の許可 ID またはアプリケーション (あるいはその両方) ごとに発行されるストアード・プロシージャーを実行します。その後、前述の方法を使用して、ストアード・プロシージャーの動作を分析します。

例: サービス・クラスによるエージェント使用の調査

ワークロード管理ソリューションでは WLM_GET_SERVICE_CLASS_AGENTS 表関数が備えられていて、この関数を用いるとサービス・クラス間におけるエージェントの相対分散を判別できます。

エージェントなどのデータ・サーバー・リソースがユーザーまたはアプリケーションのグループによって使用過多になるという状況が起こり得ます。例えば特定のユーザーのグループが、使用可能なほとんど全部のエージェントを使用していて、このグループ以外のユーザーからこの状況について知らされるとします。

実行すべき最初のステップは、各サービス・クラスを操作しているエージェント数を判別することです。以下のような照会を使用できます。

```
SELECT SUBSTR(AGENTS.SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(AGENTS.SERVICE_SUBCLASS_NAME,1,19) AS SUBCLASS_NAME,
       COUNT(*) AS AGENT_COUNT
```

```
FROM TABLE(WLM_GET_SERVICE_CLASS_AGENTS('', '', CAST(NULL AS BIGINT), -2)) AS AGENTS
WHERE AGENT_STATE = 'ACTIVE'
GROUP BY SERVICE_SUPERCLASS_NAME, SERVICE_SUBCLASS_NAME
ORDER BY SERVICE_SUPERCLASS_NAME, SERVICE_SUBCLASS_NAME

SUPERCLASS_NAME     SUBCLASS_NAME       AGENT_COUNT
------------------- ------------------- -----------
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS            7
TEST                SYSDEFAULTSUBCLASS           20
```

特定のサービス・クラスで正当な数を超えるエージェントが使用されていると判断する場合、ワークロードまたはサービス・クラスで許可されるアクティビティー数を制限するアクションを実行できます。別の方法としては、サービス・クラスに対する接続数を制限できます。

例: キャパシティー・プランニング・データが使用可能な場合のワークロード管理構成の調整

キャパシティー・プランニングを実行した場合、ユーザーのタイプおよびその予測される応答時間に関する情報があるはずです。この情報を使用して、ワークロード管理構成の構築、構成の有効性の判別、および構成の調整を行えます。

キャパシティー・プランニングを実行したと仮定し、以下の表のデータは、作業タイプおよび応答時間目標に関するその実行結果を表しているとします。

表 41. キャパシティー・プランニングの結果

作業タイプ	アプリケーション	目標	重要度	予想されるスループット
受注	orderentryapp.exe	平均応答時間を < 1 秒にする	高	1 日につき 10,000 (挿入と更新の両方)
ビジネス・インテリジェンス照会	businessobjects.exe	平均応答時間を < 10 秒にする	高	1 日につき 100 の照会
バッチ処理	batchapp.exe	スループットを最大化する	低	1 日につき 5000 の更新
その他	その他すべてのアプリケーション	ベスト・エフォート	低	1 日につき 100 のアクティビティー

前述の表のデータに基づいて、3 つのサービス・クラス (ORDER_ENTRY_SC、BI_QUERIES_SC、BATCH_SC) およびそれらのサービス・クラスに作業を割り当てる 3 つのワークロード (ORDER_ENTRY_WL、BI_QUERIES_WL、BATCH_WL) を作成できます。サービス・クラスとワークロードの作成後、統計イベント・モニターを作成して、各サービス・クラスのアクティビティー存続期間ヒストグラムなどの集約アクティビティー情報を収集できます。以下の表は、各サービス・クラスのアクティビティーの日々の平均カウント (アクティビティー存続期間ヒストグラムから算出) を、キャパシティー・プランニング実行時に予想されていた量と比較したデータであると仮定します。

表 42. 日々のアクティビティー

サービス・クラス	1 日当たりの予想されるアクティビティー	1 日当たりの実際のアクティビティー
ORDER_ENTRY_SC	10,000	9700
BI_QUERIES_SC	100	115
BATCH_SC	5000	5412
SYSDEFAULTUSERCLASS	100	85

測定されたデータは、キャパシティー・プランニングによる見積が正しかったことを示しています。以下の表は、平均アクティビティー存続期間 (アクティビティー存続期間ヒストグラムから取得) を、キャパシティー・プランニング時に判別された応答時間目標と比較したデータで、BI_QUERIES_SC サービス・クラス内のアクティビティーが応答時間の目標を満たしていないことを示しています。

表 43. 応答時間

サービス・クラス	応答時間目標	実際の平均存続期間
ORDER_ENTRY_SC	< 1 秒	0.8 秒
BI_QUERIES_SC	< 10 秒	30 秒
BATCH_SC		2 秒
SYSDEFAULTUSERCLASS		10 分

ワークロード管理インターフェースを使用すると、ビジネス・インテリジェンス照会が応答時間目標を満たさないという問題を処理する際に、以下のような様々な方法を使用できます。

- 重要度の低いサービス・クラスの並行性を制限する
- オペレーティング・システムのワークロード・マネージャーによって、重要度の低いサービス・クラスには CPU リソースがわずかしか割り当てられないようにする
- サービス・クラスのエージェントおよび入出力プリフェッチャーの優先度を変更する
- 前述の 3 つの方法を組み合わせて使用する

ビジネス・インテリジェンス照会がその目標に達しない原因となっているリソースが CPU であるとします。さらに、オペレーティング・システムのワークロード・マネージャーを使用して、SYSDEFAULTUSERCLASS サービス・クラスには他のサービス・クラスと比較して少ない CPU リソースを割り当てていると想定します。その場合、一定の日数に渡り、集約アクティビティー情報をキャプチャーして、CPU 割り振りに対して行った変更によって期待した結果が得られたかどうかを調べます。以下の表は別の比較を示しているデータで、応答時間目標と、オペレーティング・システムのワークロード・マネージャーに変更を加えた後にヒストグラムから算出された実際の平均存続期間を比較したものです。今度はすべてのサービス・クラスが応答時間の目標を達成し、CPU の再割り振りを行ったので、SYSDEFAULTUSERCLASS サービス・クラス内のアクティビティーの応答時間は 2 倍になりました。

表 44. 再構成後の応答時間

サービス・クラス	応答時間目標	実際の平均存続期間
ORDER_ENTRY_SC	< 1 秒	0.6 秒
BI_QUERIES_SC	< 10 秒	9.5 秒
BATCH_SC		1.5 秒
SYSDEFAULTUSERCLASS		20 分

例: キャパシティー・プランニング情報が使用できない場合のワークロード管理構成の調整

ワークロード管理ツールを使用すると、構成を設計するために使用するキャパシティー分析データがない場合であっても、ワークロード管理構成を設計、モニター、および調整するのに役立ちます。

次のような状況を想定してください。システムのワークロードに関して十分な知識がないか、安定した実行結果を得るために必要なワークロードをまだ知らないために、作成するワークロードおよびサービス・クラスが最初は分からないとします。また、一部のアプリケーションには応答時間要件があることは知っているものの、スピードを重要視するアプリケーションなど、他の幾つほどのアプリケーションがリソースを得るために競っているかは知らないとします。ワークロード管理モニター機能を使用すると、これが判別できます。

モニター・データを基礎として使用して、ワークロード管理構成をセットアップするには、以下のようにします。

1. 重要であると分かっているアプリケーションを分類します。そうしたアプリケーションを取り分けて、それに見合うシステム・リソースの部分を割り振ってください。
2. その他のワークロードに関しては、ワークロード内で最大のアクティビティーに関する統計を収集します。そうしたアクティビティーは、アクティビティー単位でシステムに最大の影響を与えるものだからです。
3. ステップ 2 で収集したアクティビティー情報を分析します。
4. ワークロードのまだ分類されていない部分に関して、ステップ 1 から 3 までを繰り返します。未分類の作業を分類する必要はないと判断するまで、このステップを繰り返します。

以下のセクションでは、こうしたステップを実行する方法について記します。

ステップ 1: 重要なことが分かっているアプリケーションを取り分けて、それに見合うリソースの部分を指定する

BI1 と BI2 という 2 つの重要なビジネス・インテリジェンス・アプリケーションがあり、それらのアプリケーションの応答時間を最小化する必要があると想定します。これら 2 つのアプリケーション用のワークロードを作成し、システム・リソースを割り当てることのできる MOSTIMPORTANT というサービス・クラスにそれらのワークロードをマップできます。

AIX オペレーティング・システムでは、AIX ワークロード・マネージャーを使用して MOSTIMPORTANT というサービス・クラスを作成し、このサービス・クラスに保証されたリソースの集合を割り振ることができます。

DB2 データ・サーバーでは、必要なサービス・クラスおよびワークロードを以下のようにして作成します。

```
CREATE SERVICE CLASS MOSTIMPORTANT OUTBOUND CORRELATOR 'MOSTIMPORTANT'
CREATE WORKLOAD BI1WORKLOAD APPLNAME ('BI1') SERVICE CLASS MOSTIMPORTANT
CREATE WORKLOAD BI2WORKLOAD APPLNAME ('BI2') SERVICE CLASS MOSTIMPORTANT
```

この例での意図を明白にするため、認識済みアプリケーションについて把握した後でさえ、システム・ワークロードの大部分が使途不明であるとします。そのため、このワークロードに関して理解を深め、場合によっては制御する必要があります。

ステップ 2: その他の未分類のワークロードに関して、ワークロード内の最大のアクティビティーに関する統計を収集する

長時間実行アクティビティーは、短時間実行アクティビティーに比べてシステムに与える個々の影響が大きくなります。長時間実行アクティビティーは、長期間に渡ってシステム・リソースを占有するからです。しかし、長期間実行アクティビティーに関する情報を収集しても、短時間実行アクティビティーに関する情報を収集することと比較して、オーバーヘッドが大きくなるということはありません。このため、ワークロードで最も大きな比率を占める部分に関する情報を収集する最善の方法は、上位 30% の長時間実行アクティビティーに関する情報を最初に収集することです。

アクティビティー情報を収集するアクティビティー存続期間を最初に決定して、アクティビティー情報の収集を開始します。このタスクを単純化するには、収集する未分類のアクティビティーの部分 (30 % など) を選択してから、こうしたアクティビティーのアクティビティー存続期間ヒストグラムを調査できます。システムがメモリー内統計を更新し、WLM_COLLECT_STATS プロシージャーを実行してその統計をアクティブ統計イベント・モニターに送信できるようにします。

以下の照会を使用して、SYSDEFAULTUSERCLASS サービス・クラスのアクティビティー存続期間ヒストグラムを表として取得します。その表は、各存続期間範囲に分類される合計アクティビティーの比率を表しています。この照会は、データベースがパーティション化されていないという前提で書かれています。

```
WITH TOTAL AS (
SELECT PARENTSERVICECLASSNAME,
       SERVICECLASSNAME,
       HIST.HISTOGRAM_TYPE,
       SUM(NUMBER_IN_BIN) AS NUMBER_IN_BIN
FROM HISTOGRAMBIN_DB2STATISTICS AS HIST,
     SYSCAT.SERVICECLASSES SC
WHERE
     HIST.SERVICE_CLASS_ID = SC.SERVICECLASSID
     AND HIST.TOP >= 0
     AND SC.PARENTSERVICECLASSNAME = 'SYSDEFAULTUSERCLASS'
     AND SC.SERVICECLASSNAME = 'SYSDEFAULTSUBCLASS'
     AND HIST.HISTOGRAM_TYPE = 'COORDACTLIFETIME'
GROUP BY PARENTSERVICECLASSNAME, SERVICECLASSNAME, HISTOGRAM_TYPE)
SELECT CAST(CAST(TOP AS DOUBLE) / 60000 AS DECIMAL(14,3)) AS TOP_IN_MINUTES,
       CAST(100 * CAST(SUM(HIST.NUMBER_IN_BIN) AS DOUBLE) / TOTAL.NUMBER_IN_BIN AS DECIMAL(4,2))
       AS PERCENT_IN_BIN
FROM HISTOGRAMBIN_DB2STATISTICS AS HIST,
```

```
     SYSCAT.SERVICECLASSES SC,
     TOTAL
WHERE HIST.SERVICE_CLASS_ID = SC.SERVICECLASSID
      AND HIST.TOP >= 0
      AND SC.PARENTSERVICECLASSNAME = 'SYSDEFAULTUSERCLASS'
      AND SC.SERVICECLASSNAME = 'SYSDEFAULTSUBCLASS'
      AND HIST.HISTOGRAM_TYPE = 'COORDACTLIFETIME'
      AND TOTAL.PARENTSERVICECLASSNAME = SC.PARENTSERVICECLASSNAME
      AND TOTAL.SERVICECLASSNAME = SC.SERVICECLASSNAME
      AND TOTAL.HISTOGRAM_TYPE = HIST.HISTOGRAM_TYPE
GROUP BY TOP, SC.PARENTSERVICECLASSNAME, SC.SERVICECLASSNAME, HIST.HISTOGRAM_TYPE, TOTAL.NUMBER_IN_BIN;


TOP_IN_MINUTES   PERCENT_IN_BIN
---------------- --------------
           0.000           0.00
           0.000           0.00
           0.000           0.00
           0.000           0.00
           0.000           0.00
           0.000           0.00
           0.000           0.00
           0.000           0.00
           0.000           0.00
           0.001           0.00
           0.001           0.00
           0.002           0.00
           0.004           0.00
           0.006           0.00
           0.009           0.00
           0.014           0.00
           0.021           0.00
           0.033           0.00
           0.050           0.00
           0.077           0.00
           0.118           0.00
           0.180           0.00
           0.274           0.00
           0.419           0.00
           0.639           0.00
           0.975           0.00
           1.488           0.00
           2.269           0.00
           3.462           0.00
           5.280           0.00
           8.054           0.00
          12.286           0.00
          18.740           0.00
          28.584          10.00
          43.600          15.00
          66.505          45.00
         101.442          23.00
         154.731           5.00
         236.015           2.00
         360.000           0.00
```

以下の図は、前述の照会の結果をグラフとして作図したものです。

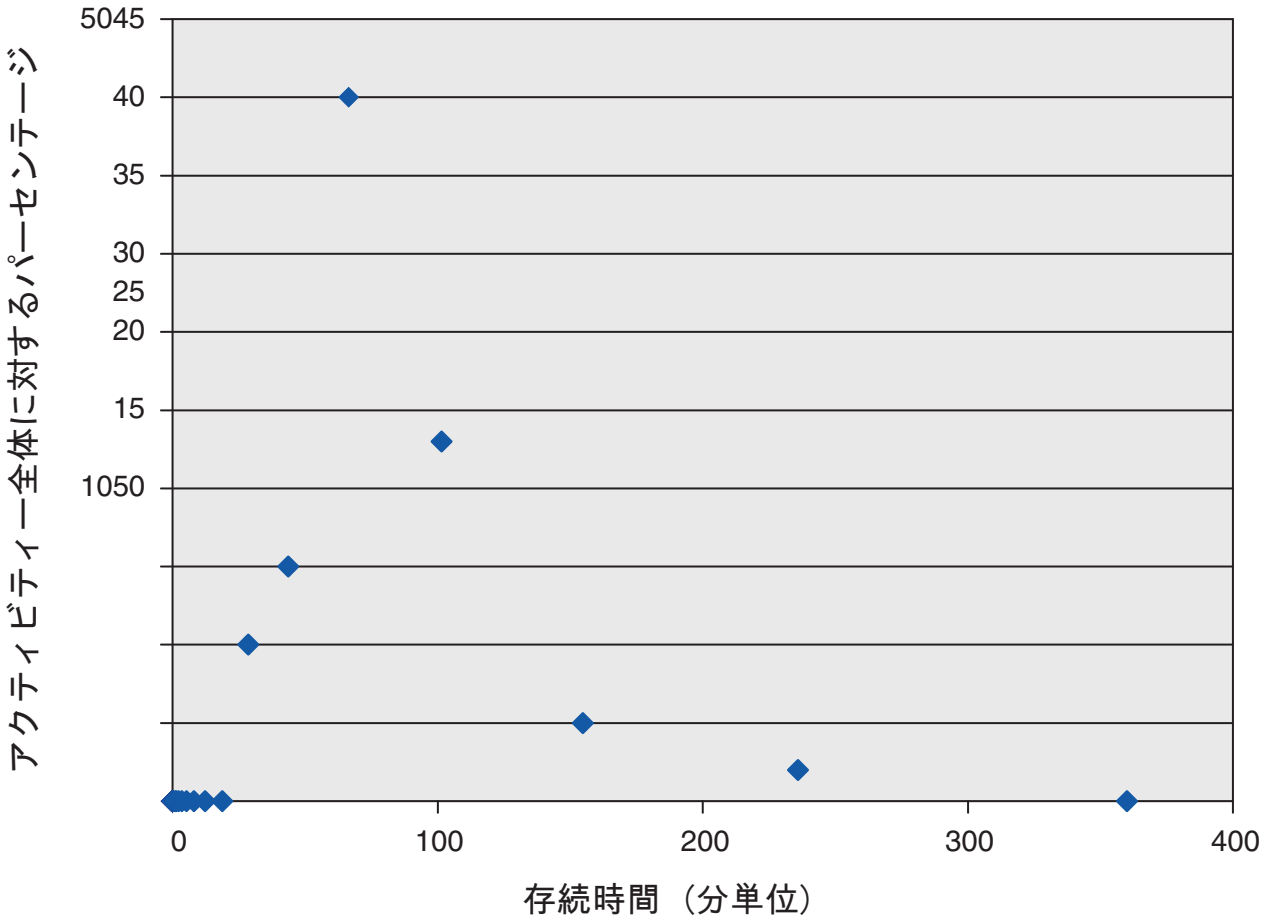


図 27. 未分類アクティビティーのアクティビティー存続期間ヒストグラム

この例では、アクティビティーの 30% が 101 分以上の存続期間範囲に該当します。こうしたアクティビティーの情報をキャプチャーするには、以下の例に示されているように CONTINUE オプションと COLLECT ACTIVITY DATA オプションを使用して、100 分のアクティビティー存続期間しきい値を作成します。このしきい値に違反すると、アクティビティー情報がアクティブ・アクティビティー・イベント・モニターに送信されます。

```
CREATE THRESHOLD COLLECTLONGESTRUNNING30PERCENT
FOR SERVICE CLASS SYSDEFAULTSUBCLASS UNDER SYSDEFAULTUSERCLASS
ACTIVITIES ENFORCEMENT DATABASE ENABLE
WHEN ACTIVITYTOTALTIME > 100 MINUTES COLLECT ACTIVITY DATA CONTINUE
```

データを収集できるようにシステムを実行します。

この 上位 30% の長時間実行アクティビティーに関する情報の収集に伴うオーバーヘッドを許容でき、数時間または数日間は引き続きデータの収集が可能であるとします。収集したデータを使用して、まだ未分類である、DML の 上位 30% の長時間実行アクティビティーを生成しているユーザーとアプリケーションを判別できます。こうしたアクティビティーには、スピードを重視するものが含まれている場合があります。優先度の低いアプリケーションがかなりの数の大きなアクティビティーを実行しているなど、予想外なことが分かることもあります。データの収集と分析が終わったら、アクティビティー存続期間に対するしきい値を削除できます。

ステップ 3: 前のステップで収集したアクティビティーに関する情報を分析する

アクティビティーをサブミットしたアプリケーション毎に、前のステップで収集したアクティビティーに関する情報を分析できます。以下の照会を指定できます。

```
SELECT SUBSTR (APPL_NAME, 1,16) APPLICATION_NAME,
       AVG(TIMESTAMPDIFF(4, CHAR(TIME_COMPLETED - TIME_CREATED)))
       AS AVG_LIFETIME_MINUTES
       COUNT(*) AS ACTIVITY_COUNT
FROM ACTIVITY_DB2ACTIVITIES
GROUP BY APPL_NAME
ORDER BY APPL_NAME

APPLICATION_NAME AVG_LIFETIME_MINUTES ACTIVITY_COUNT
================ ================== ==============
MOSTLYSMALL1                    120             21
MOSTLYSMALL2                    110             15
UNIMPORTANTAPP                  150          10213
```

サブミットしたアプリケーション毎にアクティビティーを分析すると、上位 30% の長時間実行アクティビティーの大多数は UNIMPORTANTAPP アプリケーションによってサブミットされたことがわかります。ワークロードを使用してこのアプリケーションを他の未分類のアプリケーションから分離して、BESTEFFORT というサービス・クラスにマップします。このサービス・クラスは、他のすべてのアクティビティーがそのリソースの必要性を満たした場合にのみリソースを受け取ります。

前述の結果からすると、デフォルトのサービス・クラス内の残りのアプリケーションは大規模なアクティビティーをわずかしかサブミットしていないように思われます。長時間実行アクティビティーの収集を制限せずに、デフォルトのサービス・クラス内で実行されているアクティビティーを収集するプロセスを繰り返すことが役立つ場合もあります。

ステップ 4: ワークロード内に残っている未分類の作業を分類する必要がなくなるまで、ワークロードのまだ未分類の部分に関して、ステップ 1 から 3 までを繰り返す

この時点で、2 つの重要なアプリケーションが MOSTIMPORTANT サービス・クラスで実行されていて、重要でないアプリケーションは BESTEFFORT サービス・クラスで実行され、デフォルトのユーザー・サービス・クラスではさらに重要度の低い作業が実行されています。この状況では、このサービス・クラス内のすべてのアクティビティーに関する情報を収集するのはそれほど大変なことではありません。あるいは、作業をさらに細分化する必要がなければ、ここで終了することもできます。万一、残りのワークロードに想定外の事柄が含まれている場合に備えて、そうした残りのアクティビティーに関する情報を収集できます。以下のようにして、デフォルトのユーザー・サービス・クラスに COLLECT ACTIVITY DATA を設定して、アクティビティー・イベント・モニターを作成すると、このタスクを行えます。

```
ALTER SERVICE CLASS SYSDEFAULTSUBCLASS UNDER SYSDEFAULTUSERCLASS
COLLECT ACTIVITY DATA ON COORDINATOR WITHOUT DETAILS
```

データを収集できるようにシステムを実行します。ステップ 3 の結果を分析できます。

```
SELECT SUBSTR (APPL_NAME,1,16) APPLICATION_NAME,
       AVG(TIMESTAMPDIFF(4, CHAR(TIME_COMPLETED – TIME_CREATED)))
       AS AVG_LIFETIME_MINUTES
       COUNT(*) AS ACTIVITY_COUNT
FROM ACTIVITY_DB2ACTIVITIES
GROUP BY APPL_NAME
ORDER BY APPL_NAME

APPLICATION_NAME AVG_LIFETIME_MINUTES ACTIVITY_COUNT
================ ==================== ==============
MOSTLYSMALL1                        5           1501
MOSTLYSMALL2                        7            124
ONLYSMALL                           2          10123
```

ONLYSMALL アプリケーションが未分類のアクティビティーの大多数を生成していることを、この結果は示しています。最も大規模なアクティビティーに関する情報を収集した際の結果にはこのアプリケーションが含まれていなかったので、データ収集の期間中には ONLYSMALL は大規模な照会を生成していなかったと考えることができます。

例: コストを低く見積もられた、ランタイムの高いアクティビティーの識別

次の例は、ワーク・クラス、ワーク・アクション・セット、しきい値、およびアクティビティー・コレクションを使用して、見積もりコストは低いがランタイムは高いアクティビティーを識別できる方法を示しています。このような状況は、表と索引の統計が古いために、見積もりコスト (timeron 単位) が不正確であることを示している場合があります。

最初のステップは、ワーク・クラス・セット及びワーク・クラスを作成することです。これは、見積もりコストが低いアクティビティーを識別するために使用されます。以下に例を示します。

```
CREATE WORK CLASS SET WCS1
(WORK CLASS SMALLDML WORK TYPE DML FOR TIMERONCOST FROM 0 TO 500)
```

次に、データベース・ワーク・アクション・セット及び、ワーク・アクションを作成します。これは、アクティビティー合計時間しきい値を SMALLDML ワーク・クラスに適用します。しきい値アクションは CONTINUE で、しきい値に違反したアクティビティーが完了時にアクティビティー・イベント・モニターに送信されるように、次のようにして COLLECT ACTIVITY DATA オプションを指定します。

```
CREATE WORK ACTION SET WAS1 FOR DATABASE USING WORK CLASS SET WCS1
(WORK ACTION WA1 ON WORK CLASS SMALLDML WHEN ACTIVITYTOTALTIME > 15 MINUTES
COLLECT ACTIVITY DATA WITH DETAILS CONTINUE)
```

最後に、次のようにして、しきい値違反イベント・モニターとアクティビティー・イベント・モニターを作成し、活動化します。

```
CREATE EVENT MONITOR THVIOLATIONS FOR THRESHOLD VIOLATIONS WRITE TO TABLE
SET EVENT MONITOR THVIOLATIONS STATE 1

CREATE EVENT MONITOR DB2ACTIVITIES FOR ACTIVITIES WRITE TO TABLE
SET EVENT MONITOR DB2ACTIVITIES STATE 1
```

見積もりコストが 500 timeron 未満の DML アクティビティーが 15 分を超えて実行すると、THVIOLATIONS イベント・モニターに (合計時間しきい値に違反したことを示す) しきい値違反レコードが書き込まれ、DML アクティビティーの完了時にそれについての詳細が収集されて、DB2ACTIVITIES イベント・モニターに送信さ

れます。DB2ACTIVITIES イベント・モニターのアクティビティーについて収集された情報を使用して、さらに調査を進めることができます。例えば、照会で EXPLAIN ステートメントを実行して、アクセス・プランを調べることができます。また、アクティビティーの収集時のシステム負荷とキューイングも考慮する必要があります。これは、存続時間が長いのは、システム・リソースが不足したり、アクティビティーがキューに入れられたためである可能性があるからです。存続時間が長いからといって、必ずしも統計が古いことを示しているとは限りません。

第 5 部 リファレンス

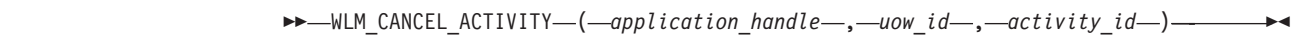
© Copyright IBM Corp. 2007, 2008

第 8 章 プロシージャーおよび表関数

WLM_CANCEL_ACTIVITY - アクティビティーのキャンセル

このプロシージャーは、指定されたアクティビティーをキャンセルします。キャンセルが行われる場合、キャンセルされたアクティビティーをサブミットしたアプリケーションにエラー・メッセージが戻されます。

構文



スキーマは SYSPROC です。

プロシージャー・パラメーター

application_handle
アクティビティーがキャンセルされるアプリケーション・ハンドルを指定する、タイプ BIGINT の入力引数。引数が NULL の場合、アクティビティーは検出されず、SQLSTATE 5U035 の SQL4702N が戻されます。

uow_id
キャンセルされるアクティビティーの作業単位 ID を指定する、タイプ INTEGER の入力引数。引数が NULL の場合、アクティビティーは検出されず、SQLSTATE 5U035 の SQL4702N が戻されます。

activity_id
キャンセルされる作業単位内のアクティビティーを一意的に識別するアクティビティー ID を指定する、タイプ INTEGER の入力引数。引数が NULL の場合、アクティビティーは検出されず、SQLSTATE 5U035 の SQL4702N が戻されます。

許可

WLM_CANCEL_ACTIVITY プロシージャーに対する EXECUTE 特権。

例

管理者は WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 表関数を使用して、アクティビティーのアプリケーション・ハンドル、作業単位 ID、およびアクティビティー ID を検索できます。アプリケーション・ハンドル 1、作業単位 ID 2、およびアクティビティー ID 3 のアクティビティーをキャンセルするには、次のようにします。

```
CALL WLM_CANCEL_ACTIVITY(1, 2, 3)
```

使用上の注意

- アクティビティーが見つからない場合、SQLSTATE 5U035 の SQL4702N が戻されます。

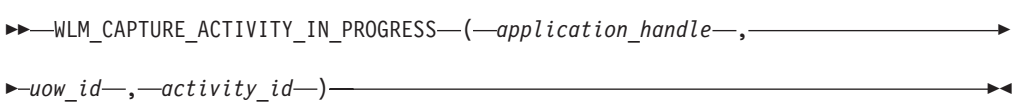
© Copyright IBM Corp. 2007, 2008

- アクティビティーが正しい状態でない (初期化されていない) ためにキャンセルできない場合、SQLSTATE 5U016 の SQL4703N (理由コード 1) が戻されます。
- アクティビティーが正常にキャンセルされた場合、SQLSTATE 57014 の SQL4725N がキャンセルされたアプリケーションに戻されます。
- キャンセル時に、コーディネーターが別のアクティビティーの要求を処理しているかまたはアイドル状態である場合、アクティビティーは CANCEL_PENDING 状態になり、コーディネーターがアクティビティーの次の要求を処理するとキャンセルされます。

WLM_CAPTURE_ACTIVITY_IN_PROGRESS - アクティビティー・イベント・モニターのアクティビティー情報の収集

このプロシージャーにより、指定したアクティビティーに関する情報が収集され、アクティブなアクティビティー・イベント・モニターに書き込まれます。このプロシージャーが子アクティビティーを持つアクティビティーに適用される場合、これは最低のレベルに至るまでの間、それぞれの子アクティビティーのレコードを再帰的に生成します。このプロシージャーが呼び出されるとすぐに、この情報が収集され、送信されます。アクティビティーが実行を完了するまで待機しません。イベント・モニター内のアクティビティーのレコードは部分レコードとしてマークが付けられます。

構文



スキーマは SYSPROC です。

プロシージャー・パラメーター

application_handle
そのアクティビティーがキャプチャーされるアプリケーション・ハンドルを指定する、タイプ BIGINT の入力引数。引数が NULL の場合、アクティビティーは検出されず、SQLSTATE 5U035 の SQL4702N が戻されます。

uow_id
キャプチャーされるアクティビティーの作業単位 ID を指定する、タイプ INTEGER の入力引数。引数が NULL の場合、アクティビティーは検出されず、SQLSTATE 5U035 の SQL4702N が戻されます。

activity_id
キャプチャーされる作業単位内のアクティビティーを一意的に識別するアクティビティー ID を指定する、タイプ INTEGER の入力引数。引数が NULL の場合、アクティビティーは検出されず、SQLSTATE 5U035 の SQL4702N が戻されます。

許可

WLM_CAPTURE_ACTIVITY_IN_PROGRESS プロシージャーに対する EXECUTE 特権。

例

特定のプロシージャー MYSCHEMA.MYSLOWSTP が通常より遅く実行しているとします。ユーザーは苦情を漏らし、管理者はスローダウンの原因の調査に乗り出します。ストアード・プロシージャーを実行しながらの調査は実際的とは言えないので、管理者はストアード・プロシージャー・アクティビティーおよびその中にネストされたアクティビティーをキャプチャーする能力を持っています。

管理者は、DB2ACTIVITIES という名前の 、既存の DB2 アクティビティーのイベント・モニターが活動化されていると想定し、WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 関数を使用して、このストアード・プロシージャーの呼び出しのアプリケーション・ハンドル、作業単位 ID、およびアクティビティー ID を取得できます。ここで、管理者がアクティビティーがアプリケーション・ハンドル 1、作業単位 ID 2、およびアクティビティー ID 3 で識別されていると想定し、WLM_CAPTURE_ACTIVITY_IN_PROGRESS への呼び出しを次のように発行できます。

```
CALL WLM_CAPTURE_ACTIVITY_IN_PROGRESS(1,2,3)
```

プロシージャーが完了すると、DB2ACTIVITIES という名前のアクティビティー・イベント・モニターについて、管理者は次の表関数を使用してアクティビティーが時間を要した場所を検出できます。

```
CREATE FUNCTION SHOWCAPTUREDACTIVITY(APPHNDL BIGINT,
                                     UOWID INTEGER,
                                     ACTIVITYID INTEGER)
  RETURNS TABLE (UOW_ID INTEGER, ACTIVITY_ID INTEGER, STMT_TEXT VARCHAR(40),
    LIFE_TIME DOUBLE)
  LANGUAGE SQL
  READS SQL DATA  NO EXTERNAL ACTION  DETERMINISTIC
  RETURN WITH RAH (LEVEL, APPL_ID, PARENT_UOW_ID, PARENT_ACTIVITY_ID,
          UOW_ID, ACTIVITY_ID, STMT_TEXT, ACT_EXEC_TIME) AS
  (SELECT 1, ROOT.APPL_ID, ROOT.PARENT_UOW_ID,
          ROOT.PARENT_ACTIVITY_ID, ROOT.UOW_ID, ROOT.ACTIVITY_ID,
          ROOTSTMT.STMT_TEXT, ACT_EXEC_TIME
   FROM ACTIVITY_DB2ACTIVITIES ROOT, ACTIVITYSTMT_DB2ACTIVITIES ROOTSTMT
   WHERE ROOT.APPL_ID = ROOTSTMT.APPL_ID AND ROOT.AGENT_ID = APPHNDL
     AND ROOT.UOW_ID = ROOTSTMT.UOW_ID AND ROOT.UOW_ID = UOWID
     AND ROOT.ACTIVITY_ID = ROOTSTMT.ACTIVITY_ID AND ROOT.ACTIVITY_ID = ACTIVITYID
  UNION ALL
   SELECT PARENT.LEVEL +1, CHILD.APPL_ID, CHILD.PARENT_UOW_ID,
          CHILD.PARENT_ACTIVITY_ID, CHILD.UOW_ID,
          CHILD.ACTIVITY_ID, CHILDSTMT.STMT_TEXT, CHILD.ACT_EXEC_TIME
   FROM RAH PARENT, ACTIVITY_DB2ACTIVITIES CHILD,
        ACTIVITYSTMT_DB2ACTIVITIES CHILDSTMT
   WHERE PARENT.APPL_ID = CHILD.APPL_ID AND
         CHILD.APPL_ID = CHILDSTMT.APPL_ID AND
         PARENT.UOW_ID = CHILD.PARENT_UOW_ID AND
         CHILD.UOW_ID = CHILDSTMT.UOW_ID AND
         PARENT.ACTIVITY_ID = CHILD.PARENT_ACTIVITY_ID AND
         CHILD.ACTIVITY_ID = CHILDSTMT.ACTIVITY_ID AND
         PARENT.LEVEL < 64
  )
```

```
SELECT UOW_ID, ACTIVITY_ID, SUBSTR(STMT_TEXT,1,40),
                ACT_EXEC_TIME AS
                LIFE_TIME
FROM RAH
```

表関数を使用する照会の例は次のとおりです。

```
SELECT * FROM TABLE(SHOWCAPTUREDACTIVITY(1, 2, 3))
  AS ACTS ORDER BY UOW_ID, ACTIVITY_ID
```

使用上の注意

アクティブなアクティビティー・イベント・モニターがない場合、SQLSTATE 01H53 の SQL1633W が戻されます。

アクティビティー情報を収集するためにこのプロシージャーを使用している場合には、入力データ値は収集されません。

WLM_COLLECT_STATS - ワークロード管理統計の収集およびリセット

このプロシージャーでは、サービス・クラス、ワークロード、作業クラス、およびしきい値キューの統計が収集され、統計イベント・モニターに書き込まれます。また、サービス・クラス、ワークロード、作業クラス、およびしきい値キューの統計がリセットされます。アクティブな統計イベント・モニターがない場合、統計のみリセットされます。

構文

スキーマは SYSPROC です。

許可

WLM_COLLECT_STATS プロシージャーに対する EXECUTE 特権。

例

例 1: WLM_COLLECT_STATS を呼び出して、統計を収集およびリセットする。

```
CALL WLM_COLLECT_STATS()
```

以下はこの照会の出力例です。

```
Return Status = 0
```

例 2: 別の呼び出しが進行中に、WLM_COLLECT_STATS を呼び出して、統計を収集およびリセットする。

```
CALL WLM_COLLECT_STATS()
```

以下はこの照会の出力例です。

```
SQL1632W The collect and reset statistics request was ignored because
another collect and reset statistics request is already in progress.
```

使用上の注意

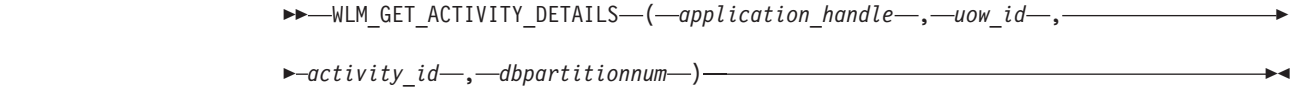
WLM_COLLECT_STATS プロシージャーは、統計を手動で収集するために使用されます。これは、WLM_COLLECT_INT データベース構成パラメーターで定義された間隔で自動的に行われるものと同じ収集 (アクティブな統計イベント・モニターへの統計の送信) およびリセット操作を実行します。プロシージャーが別の収集およびリセット要求の進行中と同時に呼び出される場合 (例えば、プロシージャーが別のプロシージャー呼び出しの実行中と同時に呼び出されたり、自動収集の発生時と同時に呼び出される)、警告 SQL1632W が SQLSTATE 01H53 とともに戻され、要求は無視されます。

WLM_COLLECT_STATS プロシージャーは、収集およびリセット・プロセスのみ開始します。このプロシージャーはプロセスが完了する前に戻ることがあります。つまり、すべての統計がアクティブな統計イベント・モニターに書き込まれる前に呼び出し元に戻すことがあります。統計の収集およびリセットが発生する速度に応じて、WLM_COLLECT_STATS プロシージャーへの呼び出し (これはアクティビティーそのもので、アクティビティー統計でカウントされる) は、前の収集間隔または開始された新規の収集間隔のいずれかでカウントされます。

WLM_GET_ACTIVITY_DETAILS - 特定のアクティビティーに関する詳細情報を戻す

この関数は、そのアプリケーション・ハンドル、作業単位 ID、およびアクティビティー ID によって識別される特定のアクティビティーに関する詳細情報を戻します。

構文



スキーマは SYSPROC です。

表関数パラメーター

application_handle
有効なアプリケーション・ハンドルを指定する、タイプ BIGINT の入力引数。引数が NULL の場合、行はこの関数から戻されません。引数が NULL の場合、SQL171N エラーが戻されます。

uow_id
アプリケーション内で固有の有効な作業単位 ID を指定する、タイプ INTEGER の入力引数。引数が NULL の場合、行はこの関数から戻されません。引数が NULL の場合、SQL171N エラーが戻されます。

activity_id
作業単位内で固有の有効なアクティビティー ID を指定する、タイプ INTEGER の入力引数。引数が NULL の場合、行はこの関数から戻されません。引数が NULL の場合、SQL171N エラーが戻されます。

dbpartitionnum
この関数を呼び出すときに現在接続されているデータベースと同じインスタンス内の有効なパーティション番号を指定する、タイプ INTEGER の入力引数。現行のデータベース・パーティションには -1、すべてのデータベース・パーティションには -2 を指定します。 NULL 値を指定すると、-1 が暗黙的に設定されます。

許可

WLM_GET_ACTIVITY_DETAILS 関数に対する EXECUTE 特権。

例

個々のアクティビティーに関する詳細情報は、WLM_GET_ACTIVITY_DETAILS 表関数を使用して取得できます。この表関数は、パーティションごとにアクティビティー情報を、名前と値のペアで戻します。この例は、アプリケーション・ハンドル 1、作業単位 ID 1、アクティビティー ID 5 で識別されるアクティビティーのパーティションごとに、名前と値のペアの 11 個のメンバー・サブセットのみを示すことに限定しています。名前と値のペアの完全なリストについては、218 ページの表 46および221 ページの表 47を参照してください。

```
SELECT SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       SUBSTR(NAME, 1, 20) AS NAME,
       SUBSTR(VALUE, 1, 30) AS VALUE
FROM TABLE(WLM_GET_ACTIVITY_DETAILS(1, 1, 5, -2)) AS ACTDETAIL
WHERE NAME IN ('APPLICATION_HANDLE',
               'COORD_PARTITION_NUM',
               'LOCAL_START_TIME',
               'UOW_ID',
               'ACTIVITY_ID',
               'PARENT_UOW_ID',
               'PARENT_ACTIVITY_ID',
               'ACTIVITY_TYPE',
               'NESTING_LEVEL',
               'INVOCATION_ID',
               'ROUTINE_ID')
ORDER BY PART
```

以下はこの照会の出力例です。

```
PART NAME                 VALUE
---- -------------------- ------------------------------
0    APPLICATION_HANDLE   1
0    COORD_PARTITION_NUM  0
0    LOCAL_START_TIME     2005-11-25-18.52.49.343000
0    UOW_ID               1
0    ACTIVITY_ID          5
0    PARENT_UOW_ID        1
0    PARENT_ACTIVITY_ID   3
0    ACTIVITY_TYPE        READ_DML
0    NESTING_LEVEL        0
0    INVOCATION_ID        1
0    ROUTINE_ID           0
1    APPLICATION_HANDLE   1
1    COORD_PARTITION_NUM  0
1    LOCAL_START_TIME     2005-11-25-18.52.49.5980001    UOW_ID               1
1    ACTIVITY_ID          5
1    PARENT_UOW_ID
1    PARENT_ACTIVITY_ID
```

```
1    ACTIVITY_TYPE        READ_DML
1    NESTING_LEVEL        0
1    INVOCATION_ID        1
1    ROUTINE_ID           0
```

使用上の注意

ACTIVITY_STATE が QUEUED である場合、コーディネーター・アクティビティーがカタログ・パーティションに対する RPC を行ってしきい値チケットを取得したが、まだ応答を受け取っていないことを意味します。この状態が表示されることは、アクティビティーが WLM によってキューに入れられていることを示すか、または短期間にわたって、アクティビティーがそのチケットを取得する処理中であることを示すことがあります。

アクティビティーが本当にキューに入れられているかどうかについてもっと正確な実態を把握するために、どのエージェントが (WLM_GET_SERVICE_CLASS_AGENTS 表関数を使用して) アクティビティーで作業しているかを判別し、カタログ・パーティションにあるこのエージェントの event_object の値が WLM_QUEUE であるかどうかを検出することができます。

戻される情報

表 45. WLM_GET_ACTIVITY_DETAILS について戻される情報

列名	データ・タイプ	説明
DBPARTITIONNUM	SMALLINT	このレコードが収集されたパーティション番号。
NAME	VARCHAR (256)	エレメント名。考えられる値については、表 46および 221 ページの表 47を参照してください。
VALUE	VARCHAR(1024)	エレメント値。考えられる値については、表 46および 221 ページの表 47を参照してください。

表 46. 戻されるエレメント

エレメント名	説明
APPLICATION_HANDLE	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。パーティションが複数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て構成されます。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。
COORD_PARTITION_NUM	アクティビティーのコーディネーター・パーティション。
UOW_ID	アプリケーション内の固有の作業単位 ID。このアクティビティーが開始した元の作業単位を表します。
ACTIVITY_ID	アプリケーション内の固有のアクティビティー ID。
PARENT_UOW_ID	アプリケーション内の固有の作業単位 ID。このアクティビティーの親アクティビティーが開始した元の作業単位を表します。アクティビティーに親アクティビティーがない場合、またはそれがリモート・パーティションにある場合は、空ストリングを戻します。
PARENT_ACTIVITY_ID	親のアクティビティー ID が ACTIVITY_ID である、作業単位内の固有のアクティビティー。アクティビティーに親アクティビティーがない場合、空ストリングを戻します。

表 46. 戻されるエレメント (続き)

エレメント名	説明
ACTIVITY_STATE	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 • 不明 (UNKNOWN)
ACTIVITY_TYPE	可能な値は以下のとおりです。 • CALL • DDL • LOAD • OTHER • READ_DML • WRITE_DML
NESTING_LEVEL	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、このアクティビティーが一番上の親アクティビティー内でネストされる深さです。
INVOCATION_ID	これは、このアクティビティーのある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します。アクティビティーがネストされていない場合にはゼロを返します。
ROUTINE_ID	ルーチン固有 ID。アクティビティーがルーチンの一部でない場合にはゼロを返します。
UTILITY_ID	アクティビティーがユーティリティーの場合、これはそのユーティリティー ID です。それ以外の場合、このフィールドは 0 です。
SERVICE_CLASS_ID	このアクティビティーが属するサービス・クラスのユニーク ID。
DATABASE_WORK_ACTION_SET_ID	このアクティビティーがデータベースに適用されている作業アクション・セットにマップされている場合、この列にはその作業アクション・セットの ID が入っています。アクティビティーがデータベースに適用されている作業アクション・セットにマップされていない場合、この列には 0 が入っています。
DATABASE_WORK_CLASS_ID	このアクティビティーがデータベースに適用されている作業アクション・セットにマップされている場合、この列にはこのアクティビティーの作業クラスの ID が入っています。アクティビティーがデータベースに適用されている作業アクション・セットにマップされていない場合、この列には 0 が入っています。

表 46. 戻されるエレメント (続き)

エレメント名	説明
SERVICE_CLASS_WORK_ACTION_SET_ID	このアクティビティーがサービス・クラスに適用されている作業アクション・セットにマップされている場合、この列にはその作業アクション・セットの ID が入っています。アクティビティーがサービス・クラスに適用されている作業アクション・セットにマップされていない場合、この列には 0 が入っています。
SERVICE_CLASS_WORK_CLASS_ID	このアクティビティーがサービス・クラスに適用されている作業アクション・セットにマップされている場合、この列にはこのアクティビティーの作業クラスの ID が入っています。アクティビティーがサービス・クラスに適用されている作業アクション・セットにマップされていない場合、この列には 0 が入っています。
ENTRY_TIME	このアクティビティーがシステムに到達した時刻。
LOCAL_START_TIME	アクティビティーがパーティションで作業を開始した時刻。これはローカル時刻です。アクティビティーがシステムに入ったが、キューに入れられており、実行を開始していない場合は、このフィールドを空ストリングにすることができます。
LAST_REFERENCE_TIME	要求がこのアクティビティーで発生するたびに、このフィールドは更新されます。
PACKAGE_NAME	アクティビティーが SQL ステートメントの場合、これはそのパッケージの名前を表します。
PACKAGE_SCHEMA	アクティビティーが SQL ステートメントの場合、これはそのパッケージのスキーマ名を表します。
PACKAGE_VERSION_ID	アクティビティーが SQL ステートメントの場合、これはそのパッケージのバージョンを表します。
SECTION_NUMBER	アクティビティーが SQL ステートメントの場合、これはそのセクション番号を表します。
STMT_PKG_CACHE_ID	ステートメント・パッケージ・キャッシュ ID。
STMT_TEXT	アクティビティーが動的 SQL であるか、またはステートメント・テキストが使用可能になっている静的 SQL である場合、このフィールドにはそのステートメント・テキストの最初の 1024 文字が入っています。そうでない場合、これは空ストリングです。
EFFECTIVE_ISOLATION	このアクティビティーに有効な分離レベル。
EFFECTIVE_LOCK_TIMEOUT	このアクティビティーに有効なロック・タイムアウト値。
EFFECTIVE_QUERY_DEGREE	このアクティビティーに有効な照会度の値。
QUERY_COST_ESTIMATE	SQL コンパイラーによって判別された照会コストの見積もり (単位は timeron)。

表 46. 戻されるエレメント (続き)

エレメント名	説明
ROWS_FETCHED	これは、表から読み取られた行の数です。これは、このレコードが記録されているデータベース・パーティションのこれらの値のみ報告します。 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。ステートメント・モニター・スイッチがオンにされていないと、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。
ROWS_MODIFIED	これは挿入、更新、または削除された行数です。これは、このレコードが記録されているデータベース・パーティションのこれらの値のみ報告します。 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。ステートメント・モニター・スイッチがオンにされていないと、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。
SYSTEM_CPU_TIME	データベース・マネージャー・エージェント・プロセス、作業単位、またはステートメントによって使用されるシステム CPU 時間の合計 (秒およびマイクロ秒)。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。
USER_CPU_TIME	データベース・マネージャー・エージェント・プロセス、作業単位、またはステートメントによって使用されるユーザー CPU 時間の合計 (秒およびマイクロ秒)。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。
QP_QUERY_ID	アクティビティーが照会の場合に、Query Patroller によってこのアクティビティーに割り当てられる照会 ID。照会 ID が 0 であると、Query Patroller が照会 ID をこのアクティビティーに割り当てなかったことを示します。

以下は、対応するしきい値がアクティビティーに適用される場合にのみ戻されます。

表 47. 適用される場合に戻されるエレメント

エレメント名	説明
CONCURRENTWORKLOADACTIVITIES_THRESHOLD_ID	しきい値の ID。
CONCURRENTWORKLOADACTIVITIES_THRESHOLD_VALUE	その値を超えると、しきい値を起動する値。
CONCURRENTWORKLOADACTIVITIES_THRESHOLD_VIOLATED	「はい」は、このアクティビティーがしきい値に違反したことを示します。「いいえ」は、このアクティビティーがしきい値に違反していないことを示します。
CONCURRENTDBCOORDACTIVITIES_DB_THRESHOLD_ID	しきい値の ID。
CONCURRENTDBCOORDACTIVITIES_DB_THRESHOLD_VALUE	その値を超えると、しきい値を起動する値。

表 47. 適用される場合に戻されるエレメント (続き)

エレメント名	説明
CONCURRENTDBCOORDACTIVITIES_DB_THRESHOLD_QUEUED	アクティビティーがこのしきい値によってキューに入れられたかどうか。
CONCURRENTDBCOORDACTIVITIES_DB_THRESHOLD_VIOLATED	「はい」は、しきい値が違反されていることを示します。「いいえ」は、しきい値が違反されていないことを示します。
CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET _THRESHOLD_ID	しきい値の ID。
CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET _THRESHOLD_VALUE	その値を超えると、しきい値を起動する値。
CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET _THRESHOLD_QUEUED	「はい」は、アクティビティーがこのしきい値によってキューに入れられたことを示します。「いいえ」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。
CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET _THRESHOLD_VIOLATED	「はい」は、しきい値が違反されていることを示します。「いいえ」は、しきい値が違反されていないことを示します。
CONCURRENTDBCOORDACTIVITIES_SUPERCLASS _THRESHOLD_ID	しきい値の ID。
CONCURRENTDBCOORDACTIVITIES_SUPERCLASS _THRESHOLD_VALUE	その値を超えると、しきい値を起動する値。
CONCURRENTDBCOORDACTIVITIES_SUPERCLASS _THRESHOLD_QUEUED	「はい」は、アクティビティーがこのしきい値によってキューに入れられたことを示します。「いいえ」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。
CONCURRENTDBCOORDACTIVITIES_SUPERCLASS _THRESHOLD_VIOLATED	「はい」は、しきい値が違反されていることを示します。「いいえ」は、しきい値が違反されていないことを示します。
CONCURRENTDBCOORDACTIVITIES_SUBCLASS _THRESHOLD_ID	しきい値の ID。
CONCURRENTDBCOORDACTIVITIES_SUBCLASS _THRESHOLD_VALUE	その値を超えると、しきい値を起動する値。
CONCURRENTDBCOORDACTIVITIES_SUBCLASS _THRESHOLD_QUEUED	アクティビティーがこのしきい値によってキューに入れられたかどうか。
CONCURRENTDBCOORDACTIVITIES_SUBCLASS _THRESHOLD_VIOLATED	「はい」は、しきい値が違反されていることを示します。「いいえ」は、しきい値が違反されていないことを示します。
ESTIMATEDSQLCOST_THRESHOLD_ID	しきい値の ID。
ESTIMATEDSQLCOST_THRESHOLD_VALUE	その値を超えると、しきい値を起動する値。
ESTIMATEDSQLCOST_THRESHOLD_VIOLATED	「はい」は、しきい値が違反されていることを示します。「いいえ」は、しきい値が違反されていないことを示します。
SQLTEMPSPACE_THRESHOLD_ID	しきい値の ID。

表 47. 適用される場合に戻されるエレメント (続き)

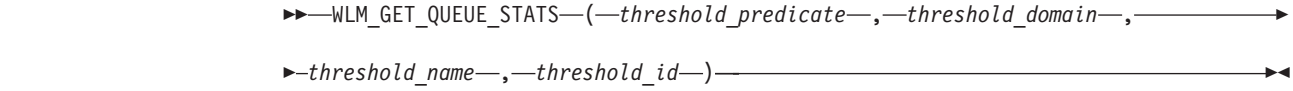
エレメント名	説明
SQLTEMPSPACE_THRESHOLD_VALUE	その値を超えると、しきい値を起動する値。
SQLTEMPSPACE_THRESHOLD_VIOLATED	「はい」は、しきい値が違反されていることを示します。「いいえ」は、しきい値が違反されていないことを示します。
SQLROWSRETURNED_THRESHOLD_ID	しきい値の ID。
SQLROWSRETURNED_THRESHOLD_VALUE	その値を超えると、しきい値を起動する値。
SQLROWSRETURNED_THRESHOLD_VIOLATED	「はい」は、しきい値が違反されていることを示します。「いいえ」は、しきい値が違反されていないことを示します。
ACTIVITYTOTALTIME_THRESHOLD_ID	しきい値の ID。
ACTIVITYTOTALTIME_THRESHOLD_VALUE	ACTIVITYTOTALTIME しきい値期間をアクティビティー開始時刻に加算して算出されるタイム・スタンプ。このタイム・スタンプに達した時にアクティビティーがまだ実行されている場合、しきい値に違反します。
ACTIVITYTOTALTIME_THRESHOLD_VIOLATED	「はい」は、しきい値が違反されていることを示します。「いいえ」は、しきい値が違反されていないことを示します。

WLM_GET_QUEUE_STATS 表関数 - しきい値キュー統計を戻す

この関数は、1 つ以上のしきい値キューの基本統計を戻します。

この関数は、しきい値キューごとに 1 行の統計を戻します。統計はすべてのアクティブ・パーティション上のキューについて戻されます。

構文



スキーマは SYSPROC です。

表関数パラメーター

threshold_predicate
有効なしきい値述部を指定する、タイプ VARCHAR(27) の入力引数。使用できる値は次のとおりです。

- CONCDBC: 並行データベース・コーディネーター・アクティビティーしきい値
- DBCONN: データベース・パーティション接続合計しきい値
- SCCONN: サービス・クラス・パーティション接続合計しきい値

- NULL または空ストリング: 考えられるすべてのしきい値述部についてデータが戻されます。 *threshold_predicate* の値は、SYSCAT.THRESHOLDS ビューの THRESHOLDPREDICATE 列の値と一致します。

threshold_domain
有効なしきい値ドメインを指定する、タイプ VARCHAR(18) の入力引数。使用できる値は次のとおりです。

- DB: データベース
- SB: サービス・サブクラス
- SP: サービス・スーパークラス
- WA: 作業アクション・セット
- NULL または空ストリング: 考えられるすべてのしきい値ドメインについてデータが戻されます。 *threshold_domain* の値は、SYSCAT.THRESHOLDS ビューの DOMAIN 列の値と一致します。

threshold_name
有効なしきい値の名前を指定する、タイプ VARCHAR(128) の入力引数。引数が NULL または空ストリングである場合、他の基準を満たすすべてのしきい値についてデータが戻されます。 *threshold_name* の値は、SYSCAT.THRESHOLDS ビューの THRESHOLDNAME 列の値と一致します。

threshold_id
有効なしきい値 ID を指定する、タイプ INTEGER の入力引数。引数が NULL または -1 である場合、他の基準を満たすすべてのしきい値についてデータが戻されます。 *threshold_id* の値は、SYSCAT.THRESHOLDS ビューの THRESHOLDID 列の値と一致します。

許可

WLM_GET_QUEUE_STATS 関数に対する EXECUTE 特権。

例

システム上のすべてのキューのすべての基本統計をすべてのパーティションの間で表示するには、次のようにします。

```
SELECT substr(THRESHOLD_NAME, 1, 6) THRESHNAME,
     THRESHOLD_PREDICATE,
     THRESHOLD_DOMAIN,
     DBPARTITIONNUM PART,
     QUEUE_SIZE_TOP,
     QUEUE_TIME_TOTAL,
     QUEUE_ASSIGNMENTS_TOTAL QUEUE_ASSIGN
FROM table(WLM_GET_QUEUE_STATS('', '', '', -1)) as QSTATS
```

以下はこの照会の出力例です。

```
THRESHNAME THRESHOLD_PREDICATE         THRESHOLD_DOMAIN   ...
---------- --------------------------- ------------------ ...
LIMIT1     CONCDBC                     DB                 ...
LIMIT2     SCCONN                      SP                 ...
LIMIT3     DBCONN                      DB                 ...
```

この照会の出力 (続き)。

```
... PART QUEUE_SIZE_TOP QUEUE_TIME_TOTAL QUEUE_ASSIGN
... ---- -------------- ---------------- ------------
... 0             12          1238540          734
... 0              4           741249           24
... 0              7           412785          128
```

使用上の注意

(パーティション上の) キューの間または (1 つ以上のキューの) パーティションの間の集約は実行されませんが、このタイプの集約は上の例で示された SQL 照会を使用して実行できます。

戻される情報

表 48. WLM_GET_QUEUE_STATS について戻される情報

列名	データ・タイプ	説明
THRESHOLD_PREDICATE	VARCHAR(27)	このキューを担当するしきい値のしきい値述部。使用できる値は次のとおりです。 • CONCDBC: 並行データベース・コーディネーター・アクティビティーしきい値 • DBCONN: データベース・パーティション接続合計しきい値 • SCCONN: サービス・クラス・パーティション接続合計しきい値 しきい値述部の値は、SYSCAT.THRESHOLDS ビューの THRESHOLDPREDICATE 列の値と一致します。
THRESHOLD_DOMAIN	VARCHAR(18)	このキューを担当するしきい値のドメイン。使用できる値は次のとおりです。 • DB: データベース • SB: サービス・サブクラス • SP: サービス・スーパークラス • WA: 作業アクション・セット しきい値ドメインの値は、SYSCAT.THRESHOLDS ビューの DOMAIN 列の値と一致します。
THRESHOLD_NAME	VARCHAR(128)	このキューに関係するしきい値の固有の名前。しきい値名前の値は、SYSCAT.THRESHOLDS ビューの THRESHOLDNAME 列の値と一致します。

表 48. WLM_GET_QUEUE_STATS について戻される情報 (続き)

列名	データ・タイプ	説明
THRESHOLD_ID	INTEGER	このキューを担当するしきい値のユニーク ID。しきい値 ID の値は、SYSCAT.THRESHOLDS ビューの THRESHOLDNAME 列の値と一致します。
DBPARTITIONNUM	SMALLINT	このレコードが収集されたパーティション番号。
SERVICE_SUPERCLASS_NAME	VARCHAR(128)	このキューを担当するしきい値のドメインであるサービス・スーパークラスの名前。しきい値のドメインがサービス・スーパークラスまたはサービス・サブクラスでない場合は NULL。
SERVICE_SUBCLASS_NAME	VARCHAR(128)	このキューを担当するしきい値のドメインであるサービス・サブクラスの名前。しきい値のドメインがサービス・サブクラスでない場合は NULL。
WORK_ACTION_SET_NAME	VARCHAR(128)	このキューを担当するしきい値のドメインである作業アクション・セットの名前。しきい値のドメインが作業アクション・セットでない場合は NULL。
WORK_CLASS_NAME	VARCHAR(128)	その作業アクションがこのキューを担当するしきい値のドメインである作業アクション・セットに属する作業クラスの名前。しきい値のドメインが作業アクション・セットでない場合は NULL。
WORKLOAD_NAME	VARCHAR(128)	このキューを担当するしきい値のドメインであるワークロードの名前。しきい値のドメインがワークロードでない場合は NULL。

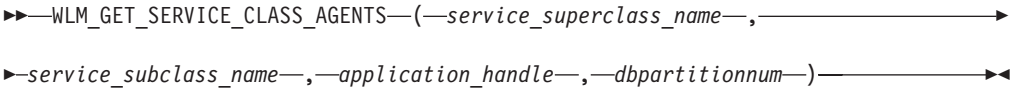
表 48. WLM_GET_QUEUE_STATS について戻される情報 (続き)

列名	データ・タイプ	説明
LAST_RESET	TIMESTAMP	統計が最後にリセットされた時刻。以下の 4 つのイベントが発生する可能性があります。これらは、統計のリセットを起動し、このタイム・スタンプを更新します。 • WLM_COLLECT_STATS プロシージャーが呼び出される。 • WLM_COLLECT_INT 構成パラメーターによって制御された定期的なコレクションおよびリセット・プロセスにより、コレクションおよびリセットが発生する。 • データベースが再活動化される。 • キュー統計が報告されているしきい値が変更され、その変更がコミットされた。 LAST_RESET タイム・スタンプがローカル時刻である。
QUEUE_SIZE_TOP	INTEGER	最後のリセット以降に達した、キュー内の接続またはアクティビティーの最高数。
QUEUE_TIME_TOTAL	BIGINT	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。
QUEUE_ASSIGNMENTS_TOTAL	BIGINT	最後のリセット以降、このキューに割り当てられた接続またはアクティビティーの数。
QUEUE_SIZE_CURRENT	INTEGER	キュー内の接続またはアクティビティーの数。
QUEUE_TIME_LATEST	BIGINT	最後の接続またはアクティビティーがキューをそのままにしておくためにキューで費やした時間。これはミリ秒単位で測定されます。
QUEUE_EXIT_TIME_LATEST	TIMESTAMP	最後の接続またはアクティビティーがキューをそのままにしておいた時間。
THRESHOLD_CURRENT_CONCURRENCY	INTEGER	しきい値に従って現在実行中の接続またはアクティビティーの数。
THRESHOLD_MAX_CONCURRENCY	INTEGER	しきい値によって現在実行中の接続またはアクティビティーの最大数。

WLM_GET_SERVICE_CLASS_AGENTS - サービス・クラスで実行中のエージェントのリスト

この関数は、指定されたサービス・クラスで実行しているか、または指定されたアプリケーションの代わりに実行している、指定されたパーティション上のエージェント、fenced モード・プロセス (db2fmps)、およびシステム・エンティティーのリストを戻します。システム・エンティティーは、非エージェント・スレッドおよびプロセス (ページ・クリーナーおよびプリフェッチャーなど) です。

構文



スキーマは SYSPROC です。

表関数パラメーター

service_superclass_name
この関数を呼び出すときに現在接続されているデータベースと同じデータベース内の有効なサービス・スーパークラス名を指定する、タイプ VARCHAR(128) の入力引数。引数が NULL または空ストリングである場合、他のパラメーターが一致する、データベース内のすべてのスーパークラスについてデータが取得されます。

service_subclass_name
スーパークラス内の特定のサブクラスを参照する、タイプ VARCHAR(128) の入力引数。引数が NULL または空ストリングである場合、他のパラメーターが一致する、データベース内のすべてのサブクラスについてデータが取得されます。

application_handle
エージェント情報を戻さなければならないアプリケーション・ハンドルを指定する、タイプ BIGINT の入力引数。引数が NULL である場合、他のパラメーターが一致する、データベース内のすべてのアプリケーションについてデータが取得されます。アプリケーション・ハンドルが 0 の場合、システム・エンティティーのみ戻されます。

dbpartitionnum
この関数を呼び出すときに現在接続されているデータベースと同じインスタンス内の有効なパーティション番号を指定する、タイプ INTEGER の入力引数。現行のデータベース・パーティションには -1、すべてのデータベース・パーティションには -2 を指定します。 NULL 値を指定すると、-1 が暗黙的に設定されます。

許可

WLM_GET_SERVICE_CLASS_AGENTS 関数に対する EXECUTE 特権。

例

すべてのデータベース・パーティションについてアプリケーション・ハンドル 1 に関連付けられたエージェントのリストを戻します。アプリケーション・ハンドルは、LIST APPLICATIONS コマンドまたは WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 表関数を使用して判別される可能性もありました。

```
SELECT SUBSTR(CHAR(APPLICATION_HANDLE),1,7) AS APPHANDLE,
     SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
     SUBSTR(CHAR(AGENT_TID),1,9) AS AGENT_TID,
     SUBSTR(AGENT_TYPE,1,11) AS AGENTTYPE,
     SUBSTR(AGENT_STATE,1,10) AS AGENTSTATE,
     SUBSTR(REQUEST_TYPE,1,12) AS REQTYPE,
     SUBSTR(CHAR(UOW_ID),1,6) AS UOW_ID,
     SUBSTR(CHAR(ACTIVITY_ID),1,6) AS ACT_ID
FROM TABLE(WLM_GET_SERVICE_CLASS_AGENTS(CAST(NULL AS VARCHAR(128)),
     CAST(NULL AS VARCHAR(128)), 1, -2)) AS SCDETAILS
ORDER BY APPHANDLE, PART, AGENT_TID
```

以下はこの照会の出力例です。

```
APPHANDLE PART AGENT_TID AGENTTYPE   AGENTSTATE REQTYPE        UOW_ID ACT_ID
--------- ---- --------- ----------- ---------- -------------- ------ ------
1         0    3         COORDINATOR ACTIVE     FETCH          1      5
1         0    4         SUBAGENT    ACTIVE     SUBSECTION:1   1      5
1         1    2         SUBAGENT    ACTIVE     SUBSECTION:2   1      5
```

ここでは、UOW ID 1 およびアクティビティー ID 5 のアクティビティーの代わりに作動しているパーティション 0 上のコーディネーター・エージェントとサブエージェントおよびパーティション 1 上のサブエージェントを示しています。コーディネーター・エージェントは要求がフェッチ要求であることを伝えています。

使用上の注意

このパラメーターの影響として、ANDing されます。つまり、サービス・スーパークラス SUP_A とサブクラス SUB_B などの競合レコードを、SUB_B が SUP_A のサブクラスにならないように指定する場合、行は戻されません。

戻される情報

表 49. WLM_GET_SERVICE_CLASS_AGENTS によって戻される情報

列名	データ・タイプ	説明
SERVICE_SUPERCLASS_NAME	VARCHAR(128)	このレコードが収集されたサービス・スーパークラスの名前。
SERVICE_SUBCLASS_NAME	VARCHAR(128)	このレコードが収集されたサービス・サブクラスの名前。
APPLICATION_HANDLE	BIGINT	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。パーティションが複数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て構成されます。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。
DBPARTITIONNUM	SMALLINT	このレコードが収集されたパーティション番号。

表 49. WLM_GET_SERVICE_CLASS_AGENTS によって戻される情報 (続き)

列名	データ・タイプ	説明
ENTITY	VARCHAR(32)	この行のエンティティーのタイプがエージェントの場合、このフィールドには ″db2agent″ が表示されます。この行のエンティティーのタイプが fenced モード・プロセスの場合、このフィールドには ″db2fmp (pid)″ が表示されます。ここで、pid は fenced モード・プロセスのプロセス ID です。それ以外の場合、システム・エンティティーの名前が表示されます。
WORKLOAD_NAME	VARCHAR(128)	このレコードが収集されたワークロードの名前。
WORKLOAD_OCCURRENCE_ID	INTEGER	ワークロード・オカレンスの ID。ワークロード・オカレンスがコーディネーター・データベース・パーティション番号およびワークロード名と結合されていなければ、これはワークロード・オカレンスを一意的に識別しません。あるいは、アプリケーション・ハンドルをコーディネーター・データベース・パーティション番号の代わりに使用することもできます。
UOW_ID	INTEGER	アプリケーション内の固有の作業単位 ID。このアクティビティーが開始した元の作業単位を表します。
ACTIVITY_ID	INTEGER	作業単位内の固有のアクティビティー ID。
PARENT_UOW_ID	INTEGER	アプリケーション内の固有の作業単位 ID。このアクティビティーの親アクティビティーが開始した元の作業単位を表します。このアクティビティーに親がない場合、NULL を戻します。
PARENT_ACTIVITY_ID	INTEGER	親のアクティビティー ID が activity_id である、作業単位内の固有のアクティビティー。このアクティビティーに親がない場合、NULL を戻します。
AGENT_TID	BIGINT	エージェントまたはシステム・エンティティーのスレッド ID。この ID が使用できない場合、このフィールドは NULL です。
AGENT_TYPE	VARCHAR(32)	コーディネーターまたはサブエージェント。コーディネーターの場合、エージェント ID はコンセントレーター環境で変わることがあります。エージェント・タイプは、以下によって表されます。 • COORDINATOR • OTHER • PDBSUBAGENT • SMPSUBAGENT
SMP_COORDINATOR	INTEGER	エージェントが smp コーディネーターかどうか。「はい」の場合は 1、「いいえ」の場合は 0。
AGENT_SUBTYPE	VARCHAR(32)	使用できるサブタイプは以下のとおりです。 • DSS • OTHER • RPC • SMP

表 49. *WLM_GET_SERVICE_CLASS_AGENTS* によって戻される情報 (続き)

列名	データ・タイプ	説明
AGENT_STATE	VARCHAR(32)	エージェントが関連付けられているか、アクティブであるかどうか。使用できる値は次のとおりです。 • ACTIVE • ASSOCIATED
EVENT_TYPE	VARCHAR(32)	このエージェントによって最後に処理されたイベントのタイプ。使用できる値は次のとおりです。 • ACQUIRE • PROCESS • WAIT
EVENT_OBJECT	VARCHAR(32)	このエージェントによって最後に処理されたイベントのオブジェクト。使用できる値は次のとおりです。 • COMPRESSION_DICTIONARY_BUILD • IMPLICIT_REBIND • INDEX_RECREATE • LOCK • LOCK_ESCALATION • QP_QUEUE • REMOTE_REQUEST • REQUEST • ROUTINE • WLM_QUEUE
EVENT_STATE	VARCHAR(32)	このエージェントによって最後に処理されたイベントの状態。使用できる値は次のとおりです。 • EXECUTING • IDLE
REQUEST_ID	VARCHAR (64)	application_handle と組み合わせる場合にのみ固有です。これは、長時間かかる 1 つの要求がある場合と複数の要求がある場合とを区別するために使用できます。例えば、複数のフェッチを 1 つの長いフェッチと区別します。

表 49. WLM_GET_SERVICE_CLASS_AGENTS によって戻される情報 (続き)

列名	データ・タイプ	説明
REQUEST_TYPE	VARCHAR(32)	要求のタイプ。使用できる値は次のとおりです。 • コーディネーター・エージェントの場合: – CLOSE – COMMIT – COMPILE – DESCRIBE – EXCSQLSET – EXECIMMD – EXECUTE – FETCH – INTERNAL <number> – OPEN – PREPARE – REBIND – REDISTRIBUTE – REORG – ROLLBACK – RUNSTATS • サブエージェント (DSS および SMP) の場合: – サブセクション番号がゼロ以外の場合に、″SUBSECTION:<subsection number>″ の形式でサブセクション番号を表示します。それ以外の場合は、NULL を戻します。

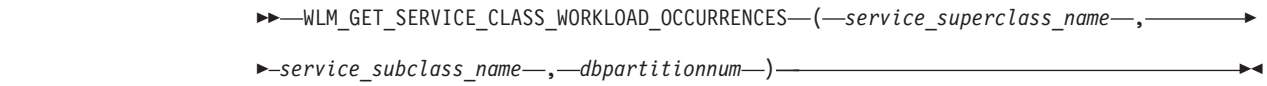
表 49. *WLM_GET_SERVICE_CLASS_AGENTS* によって戻される情報 (続き)

列名	データ・タイプ	説明
REQUEST_TYPE (続く)	VARCHAR(32)	• サブエージェント (RPC) の場合: – ABP – CATALOG – INTERNAL – REORG – RUNSTATS – WLM • サブエージェント (OTHER) の場合: – ABP – APP_RBSVPT – APP_RELSVPT – BACKUP – CLOSE – EXTERNAL_RBSVPT – EVMON – FORCE – FORCE_ALL – INTERNAL <number> – INTERRUPT – NOOP: 要求がない場合 – QP – REDISTRIBUTE – STMT_RBSVPT – STOP_USING – UPDATE_DBM_CFG – WLM 要求タイプが内部タイプのいずれかである場合、値は 'INTERNAL' の後に内部定数の実際の値が続く形で表示されます。
NESTING_LEVEL	INTEGER	これは、ID が activity_id である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、このアクティビティーが一番上の親アクティビティー内でネストされる深さです。
INVOCATION_ID	INTEGER	これは、あるアクティビティーのある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します。
ROUTINE_ID	INTEGER	ルーチン固有 ID。ルーチンの一部でない場合は NULL。

WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES - ワークロード・オカレンスのリスト

この関数は、特定のパーティション上の指定されたサービス・クラスで実行しているすべてのワークロード・オカレンスのリストを戻します。ワークロード・オカレンスとは、属性がワークロードの定義と一致しており、そのためにワークロードに関連付けられた、またはワークロードに割り当てられた特定のデータベース接続です。

構文



スキーマは SYSPROC です。

表関数パラメーター

service_superclass_name
現在接続されているデータベースで有効なサービス・スーパークラス名を指定する、タイプ VARCHAR(128) の入力引数。引数が NULL または空ストリングである場合、他のパラメーターが一致する、データベース内のすべてのスーパークラスについてデータが取得されます。

service_subclass_name
現在接続されているデータベースで有効なサービス・スーパークラス名を指定する、タイプ VARCHAR(128) の入力引数。引数が NULL または空ストリングである場合、他のパラメーターが一致する、データベース内のすべてのサブクラスについてデータが取得されます。

dbpartitionnum
現在接続されているデータベースと同じインスタンス内の有効なパーティション番号を指定する、タイプ INTEGER の入力引数。現行のデータベース・パーティションには -1、すべてのデータベース・パーティションには -2 を指示します。 NULL 値を指定すると、-1 が暗黙的に設定されます。

許可

WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 関数に対する EXECUTE 特権。

例

管理者がどのワークロード・オカレンスがシステム上で全体として実行しているかを調べたい場合、*service_superclass_name* および *service_subclass_name* NULL 値または空ストリングを指定し、 *dbpartitionnum* に -2 を指定した WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 関数を呼び出すことができます。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
```

```
       SUBSTR(CHAR(COORD_PARTITION_NUM),1,4) AS COORDPART,
       SUBSTR(CHAR(APPLICATION_HANDLE),1,7) AS APPHNDL,
       SUBSTR(WORKLOAD_NAME,1,22) AS WORKLOAD_NAME,
       SUBSTR(CHAR(WORKLOAD_OCCURRENCE_ID),1,6) AS WLO_ID
  FROM TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES
       (CAST(NULL AS VARCHAR(128)), CAST(NULL AS VARCHAR(128)), -2))
       AS SCINFO
  ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME, PART, APPHNDL,
       WORKLOAD_NAME, WLO_ID
```

システムに 4 つのデータベース・パーティションがあり、現時点で 2 つのワークロードを実行していると想定すると、上記の照会は以下のような結果を生成します。

```
SUPERCLASS_NAME     SUBCLASS_NAME      PART COORDPART ...
------------------- ------------------ ---- --------- ...
SYSDEFAULTMAINTENAN SYSDEFAULTSUBCLASS 0    0         ...
SYSDEFAULTSYSTEMCLA SYSDEFAULTSUBCLASS 0    0         ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 0    0         ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 0    0         ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 1    0         ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 1    0         ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 2    0         ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 2    0         ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 3    0         ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 3    0         ...
```

この照会の出力 (続き)。

```
... APPHNDL WORKLOAD_NAME          WLO_ID
... ------- ---------------------- ------
... -       -                      -
... -       -                      -
... 1       SYSDEFAULTUSERWORKLOAD 1
... 2       SYSDEFAULTUSERWORKLOAD 2
... 1       SYSDEFAULTUSERWORKLOAD 1
... 2       SYSDEFAULTUSERWORKLOAD 2
... 1       SYSDEFAULTUSERWORKLOAD 1
... 2       SYSDEFAULTUSERWORKLOAD 2
... 1       SYSDEFAULTUSERWORKLOAD 1
... 2       SYSDEFAULTUSERWORKLOAD 2
```

使用上の注意

このパラメーターの影響として、ANDing されます。つまり、サービス・スーパークラス SUP_A とサブクラス SUB_B などの競合レコードを、SUB_B が SUP_A のサブクラスにならないように指定する場合、行は戻されません。

注: ワークロード・オカレンスについて報告される統計 (例えば、coord_act_completed_total) が、対応するワークロード統計と結合されると、ワークロード・オカレンスについて報告される統計が各作業単位の初めにリセットされます。

戻される情報

表 50. WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES について戻される情報

列名	データ・タイプ	説明
SERVICE_SUPERCLASS_NAME	VARCHAR(128)	このレコードが収集されたサービス・スーパークラスの名前。

表 50. WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES について戻される情報 (続き)

列名	データ・タイプ	説明
SERVICE_SUBCLASS_NAME	VARCHAR(128)	このレコードが収集されたサービス・サブクラスの名前。
DBPARTITIONNUM	SMALLINT	このレコードが収集されたパーティション番号。
COORD_PARTITION_NUM	SMALLINT	指定されたワークロード・オカレンスのコーディネーター・パーティションのパーティション番号。
APPLICATION_HANDLE	BIGINT	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。パーティションが複数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て構成されます。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。
WORKLOAD_NAME	VARCHAR(128)	このレコードが収集されたワークロードの名前。
WORKLOAD_OCCURRENCE_ID	INTEGER	ワークロード・オカレンスの ID。ワークロード・オカレンスがコーディネーター・データベース・パーティション番号およびワークロード名と結合されていなければ、これはワークロード・オカレンスを一意的に識別しません。あるいは、アプリケーション・ハンドルをコーディネーター・データベース・パーティション番号の代わりに使用することもできます。

表 50. WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES について戻される情報 (続き)

列名	データ・タイプ	説明
WORKLOAD_OCCURRENCE_STATE	VARCHAR(32)	可能な値は以下のとおりです。 • DECOUPLED - ワークロード・オカレンスには割り当てられたコーディネーター・エージェントがありません (コンセントレーターの場合)。 • 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・オカレンスが強制されました。 • INTERRUPTED - ワークロード・オカレンスが中断されました。 • QUEUED - ワークロード・オカレンス・コーディネーター・エージェントが、Query Patroller またはワークロード管理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・パーティション機能 (DPF) 環境では、この状態は、コーディネーター・エージェントがカタログ・パーティションに対する RPC を行ってしきい値チケットを取得したものの、まだ応答を受け取っていないことを示す場合があります。 • TRANSIENT - ワークロード・オカレンスがまだサービス・スーパークラスにマップされていません。 • UOWEXEC - ワークロード・オカレンスは要求を処理中です。 • 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中です。
UOW_ID	INTEGER	アプリケーション内の固有の作業単位 ID。このワークロード・オカレンスが開始した元の作業単位を表します。
SYSTEM_AUTH_ID	VARCHAR(128)	ワークロード・オカレンスがシステムに注入されるときに使用したシステム許可 ID。
SESSION_AUTH_ID	VARCHAR(128)	ワークロード・オカレンスがシステムに注入されるときに使用したセッション許可 ID。
APPLICATION_NAME	VARCHAR(128)	このワークロード・オカレンスを作成したアプリケーションの名前。
CLIENT_WRKSTNNAME	VARCHAR (255)	このワークロード・オカレンスの CLIENT_WRKSTNNAME 特殊レジスターの現行値。
CLIENT_ACCTNG	VARCHAR (255)	このワークロード・オカレンスの CLIENT_ACCTNG 特殊レジスターの現行値。
CLIENT_USER	VARCHAR (255)	このワークロード・オカレンスの CLIENT_USER 特殊レジスターの現行値。

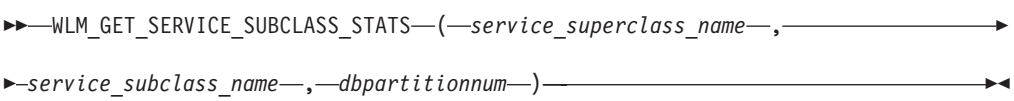
表 50. WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES について戻される情報 (続き)

列名	データ・タイプ	説明
CLIENT_APPLNAME	VARCHAR (255)	このワークロード・オカレンスの CLIENT_APPLNAME 特殊レジスターの現行値。
COORD_ACT_COMPLETED_TOTAL	INTEGER	このワークロード・オカレンスの現在の作業単位でこれまでに完了した、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数。この統計は、このワークロード・オカレンスのアクティビティーが完了し、各作業単位の初めにリセットされるたびに更新されます。
COORD_ACT_ABORTED_TOTAL	INTEGER	このワークロード・オカレンスの現在の作業単位でこれまでにアボートしたコーディネーター・アクティビティーの数。この統計は、このワークロード・オカレンスのアクティビティーがアボートされ、各作業単位の初めにリセットされるたびに更新されます。
COORD_ACT_REJECTED_TOTAL	INTEGER	このワークロード・オカレンスの現在の作業単位でこれまでにリジェクトしたコーディネーター・アクティビティーの数。アクティビティーが実行抑制作業アクションまたは予測しきい値のいずれかによって実行を妨げられている場合、そのアクティビティーはリジェクトとしてカウントされます。この統計は、このワークロード・オカレンスのアクティビティーがリジェクトされ、各作業単位の初めにリセットされるたびに更新されます。
CONCURRENT_ACT_TOP	INTEGER	現在の作業単位でこのワークロード・オカレンスについて到達している、実行状態 (アイドルおよび待機中を含む) またはキューに入れられた状態の任意のネスト・レベルの並行アクティビティーの最高数。この統計は、各作業単位の初めにリセットされます。

WLM_GET_SERVICE_SUBCLASS_STATS - サービス・サブクラスの統計を戻す

この関数は、1 つ以上のサービス・サブクラスの基本統計を戻します。

構文



スキーマは SYSPROC です。

表関数パラメーター

service_superclass_name
この関数を呼び出すときに現在接続されているデータベースと同じデータベース

内の有効なサービス・スーパークラス名を指定する、タイプ VARCHAR(128) の入力引数。引数が NULL または空ストリングである場合、データベース内のすべてのスーパークラスについてデータが取得されます。

service_subclass_name
この関数を呼び出すときに現在接続されているデータベースと同じデータベース内の有効なサービス・サブクラス名を指定する、タイプ VARCHAR(128) の入力引数。引数が NULL または空ストリングである場合、データベース内のすべてのサブクラスについてデータが取得されます。

dbpartitionnum
この関数を呼び出すときに現在接続されているデータベースと同じインスタンス内の有効なパーティション番号を指定する、タイプ INTEGER の入力引数。現行のデータベース・パーティションには -1、すべてのデータベース・パーティションには -2 を指定します。 NULL 値を指定すると、-1 が暗黙的に設定されます。

許可

EWLM_GET_SERVICE_SUBCLASS_STATS 関数に対する EXECUTE 特権。

例

例 1: すべてのアクティビティーは実行前に DB2 サービス・クラスにマップされる必要があるため、サービス・クラス統計表関数を使用し、すべてのパーティション上のすべてのサービス・クラスを照会して、システムの全体的な状態を定期的にモニターすることができます (引数に NULL 値を渡すと、最後の引数である dbpartitionnum を除いて、その引数によって結果を制限しないことを表すことに注意してください。dbpartitionnum の場合、-2 はすべてのデータベース・パーティションからのデータが戻されることを意味します。) 次のステートメントは、アクティビティー存続期間の平均および標準偏差などのサービス・クラス統計を秒単位で戻します。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       CAST(COORD_ACT_LIFETIME_AVG / 1000 AS DECIMAL(9,3))
         AS AVGLIFETIME,
       CAST(COORD_ACT_LIFETIME_STDDEV / 1000 AS DECIMAL(9,3))
         AS STDDEVLIFETIME,
       SUBSTR(CAST(LAST_RESET AS VARCHAR(30)),1,16) AS LAST_RESET
  FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS(CAST(NULL AS VARCHAR(128)),
       CAST(NULL AS VARCHAR(128)), -2)) AS SCSTATS
  ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME, PART
```

以下はこの照会の出力例です。

```
SUPERCLASS_NAME     SUBCLASS_NAME      PART ...
------------------- ------------------ ---- ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 0    ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 1    ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 2    ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 3    ...
```

この照会の出力 (続き)。

```
... AVGLIFETIME STDDEVLIFETIME LAST_RESET
... ----------- -------------- ----------------
...     691.242         34.322 2006-07-24-11.44
...     644.740         22.124 2006-07-24-11.44
...     612.431         43.347 2006-07-24-11.44
...     593.451         28.329 2006-07-24-11.44
```

例 2: また、同じ表関数が、各パーティション上のサービス・クラスで実行しているコーディネーター・アクティビティーの平均並行性の最高値を示すこともできます。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       CONCURRENT_ACT_TOP AS ACTTOP,
       CONCURRENT_WLO_TOP AS CONNTOP
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS(CAST(NULL AS VARCHAR(128)),
       CAST(NULL AS VARCHAR(128)), -2)) AS SCSTATS
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME, PART
```

以下はこの照会の出力例です。

```
SUPERCLASS_NAME     SUBCLASS_NAME      PART ACTTOP    CONNTOP
------------------- ------------------ ---- --------- ---------
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 0           10         7
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 1            0         0
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 2            0         0
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 3            0         0
```

この表関数の出力では、平均実行時間およびアクティビティーの数を調べることにより、特定のデータベースの各パーティション上の「ロード」についての優れた高水準の見解を管理者に示します。これらの表関数によって戻される高水準ゲージの大きなバリエーションは、システム上でのロードの変更を示すことがあります。

使用上の注意

対応するサービス・サブクラスの COLLECT AGGREGATE ACTIVITY DATA および COLLECT AGGREGATE REQUEST DATA 設定が NONE 以外の値に設定されている場合にのみ、一部の統計が戻されます。

WLM_GET_SERVICE_SUBCLASS_STATS 表関数は、サービス・サブクラスごとおよびパーティションごとに 1 行のデータを戻します。 (パーティション上の) サービス・クラスの間または (1 つ以上のサービス・クラスの) パーティションの間の集約は実行されません。しかし、集約は上の例で示された SQL 照会を使用して実行できます。

このパラメーターの影響として、ANDing されます。つまり、スーパークラス名 SUPA とサブクラス名 SUBB などの競合レコードを、SUBB が SUPA のサブクラスにならないように指定する場合、行は戻されません。

戻される情報

表 51. WLM_GET_SERVICE_SUBCLASS_STATS について戻される情報

列名	データ・タイプ	説明
SERVICE_SUPERCLASS_NAME	VARCHAR(128)	このレコードが収集されたサービス・スーパークラスの名前。

表 51. WLM_GET_SERVICE_SUBCLASS_STATS について戻される情報 (続き)

列名	データ・タイプ	説明
SERVICE_SUBCLASS_NAME	VARCHAR(128)	このレコードが収集されたサービス・サブクラスの名前。
DBPARTITIONNUM	SMALLINT	このレコードが収集されたパーティション番号。
LAST_RESET	TIMESTAMP	統計が最後にリセットされた時刻。以下の 4 つのイベントが発生する可能性があります。これらは、統計のリセットを起動し、このタイム・スタンプを更新します。 • WLM_COLLECT_STATS プロシージャーが呼び出される。 • WLM_COLLECT_INT 構成パラメーターによって制御された定期的なコレクションおよびリセット・プロセスにより、コレクションおよびリセットが発生する。 • データベースが再活動化される。 • 統計が報告されているサービス・サブクラスが変更され、その変更がコミットされた。 LAST_RESET タイム・スタンプがローカル時刻である。
COORD_ACT_COMPLETED_TOTAL	BIGINT	ユーザーが最後のリセット以降にサブミットし、正常に完了したコーディネーター・アクティビティーの合計数。この数は、アクティビティーが完了するたびに更新されます。
COORD_ACT_ABORTED_TOTAL	BIGINT	ユーザーが最後のリセット以降にサブミットし、エラーを出して完了したコーディネーター・アクティビティーの合計数。この数は、アクティビティーが異常終了するたびに更新されます。
COORD_ACT_REJECTED_TOTAL	BIGINT	ユーザーが最後のリセット以降にサブミットし、実行が許可される代わりに実行前に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数。アクティビティーが実行抑制作業アクションまたは予測しきい値のいずれかによって実行を妨げられている場合、そのアクティビティーはリジェクトとしてカウントされます。この数は、アクティビティーがリジェクトされるたびに更新されます。
CONCURRENT_ACT_TOP	INTEGER	このサービス・サブクラスについて到達している、実行状態 (アイドルおよび待機中を含む) またはキューに入れられた状態の任意のネスト・レベルの並行アクティビティーの最高数。

表 51. WLM_GET_SERVICE_SUBCLASS_STATS について戻される情報 (続き)

列名	データ・タイプ	説明
COORD_ACT_LIFETIME_TOP	BIGINT	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。サービス・クラスの COLLECT AGGREGATE ACTIVITY DATA が NONE の場合は、NULL です。単位はミリ秒です。
COORD_ACT_LIFETIME_AVG	DOUBLE	最後のリセット以降、このサービス・サブ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期間の算術平均。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・クラスの COLLECT AGGREGATE ACTIVITY DATA が NONE の場合は、NULL です。単位はミリ秒です。
COORD_ACT_LIFETIME_STDDEV	DOUBLE	最後のリセット以降、このサービス・サブ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期間の標準偏差。サービス・クラスの COLLECT AGGREGATE ACTIVITY DATA が NONE の場合は、NULL です。単位はミリ秒です。この標準偏差はコーディネーター・アクティビティーの存続期間ヒストグラムから計算され、ヒストグラムのサイズがデータに合わせて正しく設定されていない場合は不正確になることがあります。値が最後のヒストグラム bin に入る場合、値 -1 が戻されます。
COORD_ACT_EXEC_TIME_AVG	DOUBLE	最後のリセット以降、このサービス・サブ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・クラスの COLLECT AGGREGATE ACTIVITY DATA が NONE の場合は、NULL です。単位はミリ秒です。
COORD_ACT_EXEC_TIME_STDDEV	DOUBLE	最後のリセット以降、このサービス・サブ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標準偏差。単位はミリ秒です。この標準偏差はコーディネーター・アクティビティーの実行時間ヒストグラムから計算され、ヒストグラムのサイズがデータに合わせて正しく設定されていない場合は不正確になることがあります。値が最後のヒストグラム bin に入る場合、値 -1 が戻されます。

表 51. WLM_GET_SERVICE_SUBCLASS_STATS について戻される情報 (続き)

列名	データ・タイプ	説明
COORD_ACT_QUEUE_TIME_AVG	DOUBLE	最後のリセット以降、このサービス・サブ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・クラスの COLLECT AGGREGATE ACTIVITY DATA が NONE の場合は、NULL です。単位はミリ秒です。
COORD_ACT_QUEUE_TIME_STDDEV	DOUBLE	最後のリセット以降、このサービス・サブ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標準偏差。サービス・クラスの COLLECT AGGREGATE ACTIVITY DATA が NONE の場合は、NULL です。単位はミリ秒です。この標準偏差はコーディネーター・アクティビティーのキュー時間ヒストグラムから計算され、ヒストグラムのサイズがデータに合わせて正しく設定されていない場合は不正確になることがあります。値が最後のヒストグラム bin に入る場合、値 -1 が戻されます。
NUM_REQUESTS_ACTIVE	BIGINT	この表関数の実行時にサービス・サブクラスで実行している要求の数。
NUM_REQUESTS_TOTAL	BIGINT	最後のリセット以降、このサービス・サブクラスで実行を終了する要求の数。これは、アクティビティー内の要求のメンバーシップに関係なく、任意の要求に適用されます。このサービス・サブクラスの COLLECT AGGREGATE REQUEST DATA が NONE に設定される場合、この列の値は NULL です。
REQUEST_EXEC_TIME_AVG	DOUBLE	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。単位はミリ秒です。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。このサービス・クラスの COLLECT AGGREGATE REQUEST DATA が NONE に設定される場合、この列の値は NULL です。

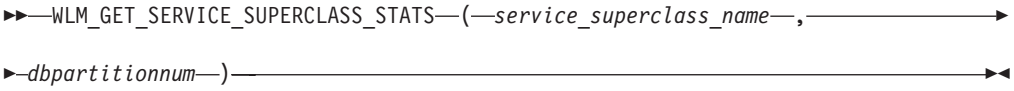
表 51. WLM_GET_SERVICE_SUBCLASS_STATS について戻される情報 (続き)

列名	データ・タイプ	説明
REQUEST_EXEC_TIME_STDDEV	DOUBLE	最後のリセット以降、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標準偏差。単位はミリ秒です。このサービス・クラスの COLLECT AGGREGATE REQUEST DATA が NONE に設定される場合、この列の値は NULL です。この標準偏差は要求実行時間ヒストグラムから計算され、ヒストグラムのサイズがデータに合わせて正しく設定されていない場合は不正確になることがあります。値が最後のヒストグラム bin に入る場合、値 -1 が戻されます。
REQUEST_EXEC_TIME__TOTAL	BIGINT	最後のリセット以降、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合計。単位はミリ秒です。このサービス・クラスの COLLECT AGGREGATE REQUEST DATA が NONE に設定される場合、この列の値は NULL です。

WLM_GET_SERVICE_SUPERCLASS_STATS - サービス・スーパークラスの統計を戻す

この関数は、1 つ以上のサービス・スーパークラスの基本統計を戻します。

構文



スキーマは SYSPROC です。

表関数パラメーター

service_superclass_name
この関数を呼び出すときに現在接続されているデータベースと同じデータベース内の有効なサービス・スーパークラス名を指定する、タイプ VARCHAR(128) の入力引数。引数が NULL または空ストリングである場合、データベース内のすべてのスーパークラスについてデータが取得されます。

dbpartitionnum
この関数を呼び出すときに現在接続されているデータベースと同じインスタンス内の有効なパーティション番号を指定する、タイプ INTEGER の入力引数。現行のデータベース・パーティションには -1、すべてのデータベース・パーティションには -2 を指定します。 NULL 値を指定すると、-1 が暗黙的に設定されます。

許可

WLM_GET_SERVICE_SUPERCLASS_STATS 関数に対する EXECUTE 特権。

例

システム上のすべてのサービス・スーパークラスのすべての基本統計をすべてのデータベース・パーティションの間で表示するには、次のようにします。

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME, 1, 26) SERVICE_SUPERCLASS_NAME,
       DBPARTITIONNUM,
       LAST_RESET,
       CONCURRENT_CONNECTION_TOP CONCURRENT_CONN_TOP
FROM TABLE(WLM_GET_SERVICE_SUPERCLASS_STATS('', -2)) as SCSTATS
```

以下はこの照会の出力例です。

```
SERVICE_SUPERCLASS_NAME    DBPARTITIONNUM ...
-------------------------- -------------- ...
SYSDEFAULTSYSTEMCLASS                   0 ...
SYSDEFAULTMAINTENANCECLASS              0 ...
SYSDEFAULTUSERCLASS                     0 ...
```

この照会の出力 (続き)。

```
... LAST_RESET                 CONCURRENT_CONN_TOP
... -------------------------- -------------------
... 2006-09-05-09.38.44.396788                   0
... 2006-09-05-09.38.44.396795                   0
... 2006-09-05-09.38.44.396796                   1
```

使用上の注意

WLM_GET_SERVICE_SUPERCLASS_STATS 表関数は、サービス・スーパークラスおよびパーティションごとに1 行のデータを戻します。 (パーティション上の) サービス・スーパークラスの間または (1 つ以上のサービス・スーパークラスの) パーティションの間の集約は実行されません。しかし、集約は上の例で示された SQL 照会を使用して実行できます。

戻される情報

表 52. WLM_GET_SERVICE_SUPERCLASS_STATS について戻される情報

列名	データ・タイプ	説明
SERVICE_SUPERCLASS_NAME	VARCHAR(128)	このレコードが収集されたサービス・スーパークラスの名前。
DBPARTITIONNUM	SMALLINT	このレコードが収集されたパーティション番号。

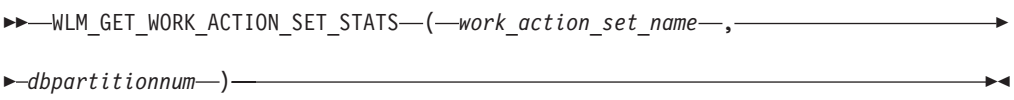
表 52. *WLM_GET_SERVICE_SUPERCLASS_STATS* について戻される情報 (続き)

列名	データ・タイプ	説明
LAST_RESET	TIMESTAMP	統計が最後にリセットされた時刻。以下の 4 つのイベントが発生する可能性があります。これらは、統計のリセットを起動し、このタイム・スタンプを更新します。 • WLM_COLLECT_STATS プロシージャーが呼び出される。 • WLM_COLLECT_INT 構成パラメーターによって制御された定期的なコレクションおよびリセット・プロセスにより、コレクションおよびリセットが発生する。 • データベースが再活動化される。 • 統計が報告されているサービス・スーパークラスが変更され、その変更がコミットされた。 LAST_RESET タイム・スタンプがローカル時刻である。
CONCURRENT_CONNECTION_TOP	INTEGER	最後のリセット以降にこのクラスで達した並行コーディネーター接続の最高数。

WLM_GET_WORK_ACTION_SET_STATS - 作業アクション・セット統計を戻す

この関数は、作業アクション・セットの統計を戻します。

構文



スキーマは SYSPROC です。

表関数パラメーター

work_action_set_name
統計を戻す特定の作業アクション・セットを指定する、タイプ VARCHAR(128) の入力引数。引数が NULL または空ストリングである場合、すべての作業アクション・セットについて統計が戻されます。

dbpartitionnum
この関数を呼び出すときに現在接続されているデータベースと同じインスタンス内の有効なパーティション番号を指定する、タイプ INTEGER の入力引数。現行のデータベース・パーティションには -1、すべてのデータベース・パーティションには -2 を指定します。 NULL 値を指定すると、-1 が暗黙的に設定されます。

許可

WLM_GET_WORK_ACTION_SET_STATS 関数に対する EXECUTE 特権。

例

3 つの作業クラス、ReadClass、WriteClass、および LoadClass があると想定します。 ReadClass に関連した作業アクションと LoadClass に関連した作業アクションはありますが、WriteClass に関連した作業アクションはありません。パーティション 0 では、ReadClass で現在実行中の (またはキューに入れられた) 8 つのアクティビティー、WriteClass で現在実行中の (またはキューに入れられた) 4 つのアクティビティー、LoadClass で現在実行中の (またはキューに入れられた) 2 つのアクティビティー、およびどの作業クラスにも割り当てられていない現在実行中の (またはキューに入れられた) 3 つのアクティビティーがあります。 WriteClass 作業クラスに関連した作業アクションはないため、その作業クラスが該当する 4 つのアクティビティーは、どの作業クラスにも割り当てられなかった 3 つのアクティビティーとともに人工的な“*” クラスでカウントされます。

```
SELECT SUBSTR(WORK_ACTION_SET_NAME,1,18) AS WORK_ACTION_SET_NAME,
     SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
     SUBSTR(WORK_CLASS_NAME,1,15) AS WORK_CLASS_NAME,
     LAST_RESET,
     SUBSTR(CHAR(WLO_ACT_TOTAL),1,14) AS TOTAL_WLO_ACTS
FROM TABLE(WLM_GET_WORK_ACTION_SET_STATS
     (CAST(NULL AS VARCHAR(128)), -2)) AS WASSTATS
ORDER BY WORK_ACTION_SET_NAME, WORK_CLASS_NAME, PART
```

以下はこの照会の出力例です。

```
WORK_ACTION_SET_NAME PART WORK_CLASS_NAME LAST_RESET                TOTAL_WLO_ACTS
-------------------- ---- --------------- -------------------------- --------------
AdminActionSet       0    ReadClass       2005-11-25-18.52.49.343000 8
AdminActionSet       1    ReadClass       2005-11-25-18.52.50.478000 0
AdminActionSet       0    LoadClass       2005-11-25-18.52.49.343000 2
AdminActionSet       1    LoadClass       2005-11-25-18.52.50.478000 0
AdminActionSet       0    *               2005-11-25-18.52.49.343000 7
AdminActionSet       1    *               2005-11-25-18.52.50.478000 0
```

戻される情報

表 53. WLM_GET_WORK_ACTION_SET_STATS について戻される情報

列名	データ・タイプ	説明
WORK_ACTION_SET_NAME	VARCHAR(128)	アクション・セットの名前。
DBPARTITIONNUM	SMALLINT	このレコードが収集されたパーティション番号。

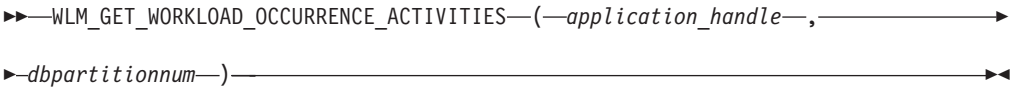
表 53. WLM_GET_WORK_ACTION_SET_STATS について戻される情報 (続き)

列名	データ・タイプ	説明
LAST_RESET	TIMESTAMP	統計が最後にリセットされた時刻。以下の 4 つのイベントが発生する可能性があります。これらは、統計のリセットを起動し、このタイム・スタンプを更新します。 • WLM_COLLECT_STATS プロシージャーが呼び出される。 • WLM_COLLECT_INT 構成パラメーターによって制御された定期的なコレクションおよびリセット・プロセスにより、コレクションおよびリセットが発生する。 • データベースが再活動化される。 • 統計が報告されている作業アクション・セットが変更され、その変更がコミットされた。 LAST_RESET タイム・スタンプがローカル時刻である。
WORK_CLASS_NAME	VARCHAR(128)	指定された作業アクション・セットに関連した作業クラスの名前。作業クラスをこの表に表示するには、この作業クラスに関連付けられた作業アクションがなければなりません。 “*” は、ユーザーが 1 つ以上の作業アクションを関連付けたその他の作業クラスに属さない、すべてのアクティビティーをカウントするために作成された人工的な作業クラスを表します。
ACT_TOTAL	BIGINT	WORK_CLASS_NAME で指定される作業クラスに割り当てられた、ネスト・レベルのアクティビティーの数。

WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES - アクティビティーのリストを戻す

この関数は、指定されたパーティション上の特定のアプリケーションからサブミットされ、また完了していないすべてのアクティビティーのリストを戻します。

構文



スキーマは SYSPROC です。

表関数パラメーター

application_handle
アクティビティーのリストが戻されるアプリケーション・ハンドルを指定する、タイプ BIGINT の入力引数。引数が NULL である場合、他のパラメーターが一致する、データベース内のすべてのアプリケーションについてデータが取得されます。

dbpartitionnum
この関数を呼び出すときに現在接続されているデータベースと同じインスタンス

内の有効なパーティション番号を指定する、タイプ INTEGER の入力引数。現行のデータベース・パーティションには -1、すべてのデータベース・パーティションには -2 を指定します。 NULL 値を指定すると、-1 が暗黙的に指定されます。

許可

WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES 関数に対する EXECUTE 特権。

例

アプリケーション・ハンドルが識別されたら、このアプリケーションで現在実行中のすべてのアクティビティーを検索できます。例えば、管理者が list applications コマンドを使用して判別したアプリケーション・ハンドルが 1 であることがわかったアプリケーションのアクティビティーをリストするとします。

```
SELECT SUBSTR(CHAR(COORD_PARTITION_NUM),1,5) AS COORD,
     SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
     SUBSTR(CHAR(UOW_ID),1,5) AS UOWID,
     SUBSTR(CHAR(ACTIVITY_ID),1,5) AS ACTID,
     SUBSTR(CHAR(PARENT_UOW_ID),1,8) AS PARUOWID,
     SUBSTR(CHAR(PARENT_ACTIVITY_ID),1,8) AS PARACTID,
            ACTIVITY_TYPE AS ACTTYPE,
     SUBSTR(CHAR(NESTING_LEVEL),1,7) AS NESTING
FROM TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(1, -2)) AS WLOACTS
ORDER BY PART, UOWID, ACTID
```

以下はこの照会の出力例です。

```
COORD PART UOWID ACTID PARUOWID PARACTID ACTTYPE  NESTING
----- ---- ----- ----- -------- -------- -------- -------
0     0    2     3     -        -        CALL     0
0     0    2     5     2        3        READ_DML 1
```

戻される情報

表 54. WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES によって戻される情報

列名	データ・タイプ	説明
APPLICATION_HANDLE	BIGINT	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。パーティションが複数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て構成されます。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。
DBPARTITIONNUM	SMALLINT	このレコードが収集されたパーティション番号。
COORD_PARTITION_NUM	SMALLINT	アクティビティーのコーディネーター・パーティション。
LOCAL_START_TIME	TIMESTAMP	アクティビティーがパーティションで作業を開始した時刻。これはローカル時刻です。アクティビティーがシステムに入ったが、キューに入れられており、実行を開始していない場合は、このフィールドを NULL にすることができます。

表 54. WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES によって戻される情報 (続き)

列名	データ・タイプ	説明
UOW_ID	INTEGER	アプリケーション内の固有の作業単位 ID。アクティビティーが開始した元の作業単位を表します。
ACTIVITY_ID	INTEGER	作業単位内の固有のアクティビティー ID。
PARENT_UOW_ID	INTEGER	アプリケーション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開始した元の作業単位を表します。アクティビティーに親アクティビティーがない場合、またはそれがリモート・パーティションにある場合は、NULL を戻します。
PARENT_ACTIVITY_ID	INTEGER	親のアクティビティー ID が ACTIVITY_ID である、作業単位内の固有のアクティビティー。アクティビティーに親アクティビティーがない場合、またはそれがリモート・パーティションにある場合は、NULL を戻します。

表 54. WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES によって戻される情報 (続き)

列名	データ・タイプ	説明
ACTIVITY_STATE	VARCHAR(32)	可能な値は以下のとおりです。 • CANCEL_PENDING - アクティビティーの要求をアクティブに実行しているエージェントがないときに、アクティビティーは取り消されました。次に要求がアクティビティーの一部としてサブミットされるときに、そのアクティビティーは取り消され、そのアクティビティーをサブミットしたユーザーは SQL4725N エラーを受け取ります。 • EXECUTING - エージェントはアクティビティーの要求をアクティブに実行しています。 • IDLE - アクティビティーの要求をアクティブに処理しているエージェントがありません。 • INITIALIZING - アクティビティーはサブミットされましたが、まだ実行を開始していません。初期化状態中に、予測しきい値がアクティビティーに適用され、アクティビティーが実行を許可されるかどうかを判別します。 • QP_CANCEL_PENDING - CANCEL_PENDING 状態と同じですが、アクティビティーは WLM_CANCEL_ACTIVITY プロシージャーではなく、Query Patroller によって取り消されました。 • QP_QUEUED - アクティビティーは Query Patroller によってキューに入れられます。 • QUEUED - アクティビティーが、ワークロード管理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・パーティション機能 (DPF) 環境では、この状態は、コーディネーター・エージェントがカタログ・パーティションに対する RPC を行ってしきい値チケットを取得したものの、まだ応答を受け取っていないことを意味する場合があります。この状態が表示されることは、アクティビティーがワークロード管理キューイングしきい値によってキューに入れられていることを示すか、または短期間にわたって、アクティビティーがそのチケットを取得する処理中であることを示すことがあります。アクティビティーが本当にキューに入れられているかどうかについてもっと正確な実態を把握するために、どのエージェントがアクティビティーで作業しているかを判別し、カタログ・パーティションにあるこのエージェントの EVENT_OBJECT の値が WLM_QUEUE であるかどうかを検出することができます。 • TERMINATING - アクティビティーは実行を完了し、システムから除去されています。

表 54. WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES によって戻される情報 (続き)

列名	データ・タイプ	説明
ACTIVITY_TYPE	VARCHAR(32)	可能な値は以下のとおりです。 • CALL • DDL • LOAD • OTHER • READ_DML • WRITE_DML 各アクティビティー・タイプに関連付けられた SQL ステートメントの様々なタイプの説明については、ワークロード・マネージャー ガイドおよびリファレンスの『作業クラスの作業タイプおよび SQL ステートメント』を参照してください。
NESTING_LEVEL	INTEGER	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、このアクティビティーが一番上の親アクティビティー内でネストされる深さです。
INVOCATION_ID	INTEGER	これは、このアクティビティーのある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します。
ROUTINE_ID	INTEGER	ルーチン固有 ID。
UTILITY_ID	INTEGER	アクティビティーがユーティリティーの場合、これはそのユーティリティー ID です。それ以外の場合、このフィールドは NULL です。
SERVICE_CLASS_ID	INTEGER	このアクティビティーが属するサービス・クラスのユニーク ID。
DATABASE_WORK_ACTION_SET_ID	INTEGER	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分類されている場合、この列にはこの作業クラスがメンバーとなっている作業クラス・セットの ID が入っています。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分類されていない場合、この列には NULL が入っています。
DATABASE_WORK_CLASS_ID	INTEGER	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分類されている場合、この列には作業クラスの ID が入っています。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分類されていない場合、この列には NULL が入っています。
SERVICE_CLASS_WORK_ACTION_SET_ID	INTEGER	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分類されている場合、この列には作業クラスが属する作業クラス・セットに関連付けられた作業アクション・セットの ID が入っています。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分類されていない場合、この列には NULL が入っています。

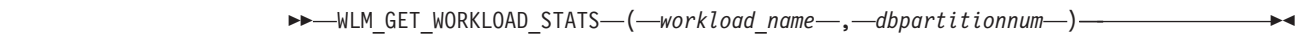
表 54. *WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES* によって戻される情報 (続き)

列名	データ・タイプ	説明
SERVICE_CLASS_WORK_CLASS_ID	INTEGER	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分類されている場合、この列に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 が入っています。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分類されていない場合、この列には NULL が入っています。

WLM_GET_WORKLOAD_STATS - ワークロード統計を戻す

この関数は、ワークロード名とデータベース・パーティション番号のすべての組み合わせについてのワークロード統計を戻します。

構文



スキーマは SYSPROC です。

表関数パラメーター

workload_name
統計が戻される特定のワークロードを指定する、タイプ VARCHAR(128) の入力引数。引数が NULL または空ストリングである場合、すべてのワークロードについて統計が戻されます。

dbpartitionnum
この関数を呼び出すときに現在接続されているデータベースと同じインスタンス内の有効なパーティション番号を指定する、タイプ INTEGER の入力引数。現行のデータベース・パーティションには -1、すべてのデータベース・パーティションには -2 を指定します。 NULL 値を指定すると、-1 が暗黙的に設定されます。

許可

WLM_GET_WORKLOAD_STATS 関数に対する EXECUTE 特権。

例

管理者がワークロードの統計を調べるとします。これは以下の照会を使用して行うことができます。

```
SELECT SUBSTR(WORKLOAD_NAME,1,22) AS WL_DEF_NAME,
     SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
     CONCURRENT_WLO_TOP AS WLO_TOP,
     CONCURRENT_WLO_ACT_TOP AS WLO_ACT_TOP
FROM TABLE(WLM_GET_WORKLOAD_STATS(CAST(NULL AS VARCHAR(128)), -2))
  AS WLSTATS
ORDER BY WL_DEF_NAME, PART
```

以下はこの照会の出力例です。

```
WL_DEF_NAME            PART WLO_TOP         WLO_ACT_TOP
---------------------- ---- --------------- -------------------
MYUSERWORKLOAD         0                 2                   8
MYUSERWORKLOAD         1                 0                   0
SYSDEFAULTUSERWORKLOAD 0                 1                   1
SYSDEFAULTUSERWORKLOAD 1                 0                   0
```

ここで、パーティション 0 では、MYUSERWORKLOAD ワークロードの並行オカレンスの最高数が 2 であり、これらのワークロード・オカレンスのいずれかの並行アクティビティーの最高数が 8 であることがわかります。

使用上の注意

この関数は、ワークロード名とデータベース・パーティション番号のすべての組み合わせについて 1 行を戻します。ワークロードの間、パーティションの間、またはサービス・クラスの間の集約は実行されません。しかし、集約は SQL 照会を使用して実行できます。

戻される情報

表 55. WLM_GET_WORKLOAD_STATS によって戻される情報

列名	データ・タイプ	説明
WORKLOAD_NAME	BIGINT	このレコードが収集されたワークロードの名前。
DBPARTITIONNUM	SMALLINT	このレコードが収集されたパーティション番号。
LAST_RESET	TIMESTAMP	統計が最後にリセットされた時刻。以下の 4 つのイベントが発生する可能性があります。これらは、統計のリセットを起動し、このタイム・スタンプを更新します。 • WLM_COLLECT_STATS プロシージャーが呼び出される。 • WLM_COLLECT_INT 構成パラメーターによって制御された定期的なコレクションおよびリセット・プロセスにより、コレクションおよびリセットが発生する。 • データベースが再活動化される。 • 統計が報告されているワークロードが変更され、その変更がコミットされた。 LAST_RESET タイム・スタンプがローカル時刻である。
CONCURRENT_WLO_TOP	INTEGER	最後のリセット以降、このパーティション上の指定されたワークロードの並行オカレンスの最高数。
CONCURRENT_WLO_ACT_TOP	INTEGER	最後のリセット以降、このワークロードのいずれかのオカレンスで到達した、実行状態 (アイドルおよび待機中を含む) またはキューに入れられた状態の並行アクティビティー (コーディネーターとネストを含む) の最高数。各ワークロード・オカレンスによってその作業単位の終わりに更新されます。

表 55. WLM_GET_WORKLOAD_STATS によって戻される情報 (続き)

列名	データ・タイプ	説明
COORD_ACT_COMPLETED_TOTAL	BIGINT	最後のリセット以降に完了したこのワークロードのいずれかのオカレンスに割り当てられた、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。各ワークロード・オカレンスによってその作業単位の終わりに更新されます。
COORD_ACT_ABORTED_TOTAL	BIGINT	最後のリセット以降で完了前にアボートされたこのワークロードのいずれかのオカレンスに割り当てられた、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。各ワークロード・オカレンスによってその作業単位の終わりに更新されます。
COORD_ACT_REJECTED_TOTAL	BIGINT	最後のリセット以降で実行前にリジェクトされたこのワークロードのいずれかのオカレンスに割り当てられた、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。各ワークロード・オカレンスによってその作業単位の終わりに更新されます。アクティビティーが実行抑制作業アクションまたは予測しきい値のいずれかによって実行を妨げられている場合、そのアクティビティーはリジェクトとしてカウントされます。 WLM_GET_SERVICE_SUBCLASS_STATS 関数の同じ名前の列とは異なり、これはアクティビティーがサービス・クラスに割り当てられる前に発生するリジェクトもカウントします。アクティビティーが ConcurrentWorkloadOccurrences しきい値に違反すると、そうしたリジェクトの例が発生します。
WLO_COMPLETED_TOTAL	BIGINT	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。

第 9 章 モニター・エレメント

ワークロード管理に関するモニター・エレメント

次のモニター・エレメントにより、アクティビティー、しきい値違反、およびワークロード管理の統計に関する情報が提供されます。

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。

エレメント ID

activate_timestamp

エレメント・タイプ

タイム・スタンプ

表 56. イベント・モニター情報

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。

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。

エレメント ID

activity_collected

エレメント・タイプ

情報

表 57. イベント・モニター情報

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	-

© Copyright IBM Corp. 2007, 2008

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。アクティビティー・イベント・モニター・レコード中でこのモニター・エレメントを **appl_id** および **uow_id** と一緒に使用すると、収集されたアクティビティーを一意的に識別します。しきい値違反イベント・モニター・レコード中でこのモニター・エレメントを **appl_id** および **uow_id** と一緒に使用すると、しきい値に違反したアクティビティーを一意的に識別します。

エレメント ID

activity_id

エレメント・タイプ

情報

表 58. イベント・モニター情報

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 **uow_id** および **agent_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。

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。

エレメント ID

activity_secondary_id

エレメント・タイプ

情報

表 59. イベント・モニター情報

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_type アクティビティー・タイプ : モニター・エレメント

このアクティビティー・レコードが適用されるアクティビティーのタイプ。

エレメント ID

activity_type

エレメント・タイプ

情報

表 60. イベント・モニター情報

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

使用できる値は次のとおりです。

- LOAD
- READ_DML
- WRITE_DML
- DDL
- CALL

- OTHER

リモート・パーティションでは、このモニター・エレメントの値は常に OTHER です。

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。

エレメント ID

act_exec_time

エレメント・タイプ

時間

表 61. イベント・モニター情報

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_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。

エレメント ID

act_total

エレメント・タイプ

カウンター

表 62. イベント・モニター情報

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。

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。

エレメント ID

arm_correlator

エレメント・タイプ

情報

表 63. イベント・モニター情報

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。

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。

エレメント ID

bin_id

エレメント・タイプ

情報

表 64. イベント・モニター情報

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。

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。

エレメント ID

bottom

エレメント・タイプ

情報

表 65. イベント・モニター情報

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。

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。

エレメント ID

concurrent_act_top

エレメント・タイプ

水準点

表 66. イベント・モニター情報

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。

エレメント ID

concurrent_connection_top

エレメント・タイプ

水準点

表 67. イベント・モニター情報

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。

エレメント ID

concurrent_wlo_act_top

エレメント・タイプ

水準点

表 68. イベント・モニター情報

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。

エレメント ID

concurrent_wlo_top

エレメント・タイプ

水準点

表 69. イベント・モニター情報

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。

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。

エレメント ID

coord_act_aborted_total

エレメント・タイプ

カウンター

表 70. イベント・モニター情報

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。

エレメント ID

coord_act_completed_total

エレメント・タイプ

カウンター

表 71. イベント・モニター情報

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_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。

エレメント ID

coord_act_lifetime_top

エレメント・タイプ

水準点

表 72. イベント・モニター情報

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	-

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_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。

エレメント ID

coord_act_rejected_total

エレメント・タイプ

カウンター

表 73. イベント・モニター情報

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_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

このアクティビティーのコーディネーター・パーティションのパーティション番号

エレメント ID

coord_partition_num

エレメント・タイプ

情報

表 74. イベント・モニター情報

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	-
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の、コーディネーター・パーティションを識別できます。

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。

エレメント ID

cost_estimate_top

エレメント・タイプ

水準点

表 75. イベント・モニター情報

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	-

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。

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。単位はミリ秒です。

エレメント ID

coord_act_lifetime_avg

エレメント・タイプ

情報

表 76. イベント・モニター情報

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	-

使用法

この統計を使用すると、完了またはアボートしたサービス・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_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。単位はミリ秒です。最後のリセット以降に完了またはアボートしたこのサービス・サブクラスに 0 が関連付けられます。サービス・クラスの COLLECT AGGREGATE ACTIVITY DATA が NONE の場合は -1 が返されます。単位はミリ秒です。

エレメント ID

coord_act_queue_time_avg

エレメント・タイプ

情報

表 77. イベント・モニター情報

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	-

使用法

この統計を使用すると、完了またはアボートしたサービス・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_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。単位はミリ秒です。

エレメント ID

coord_act_exec_time_avg

エレメント・タイプ

情報

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
統計	event_scstats	-
統計	event_wcstats	-

使用法

この統計を使用すると、完了またはアボートしたサービス・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。

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。

エレメント ID

request_exec_time_avg

エレメント・タイプ

情報

表 79. イベント・モニター情報

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。

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。単位はミリ秒です。

エレメント ID

coord_act_est_cost_avg

エレメント・タイプ

情報

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	-
統計	event_wcstats	-

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アボートしたこのサービス・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_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。単位はミリ秒です。

エレメント ID

coord_act_interarrival_time_avg

エレメント・タイプ

情報

表 81. イベント・モニター情報

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	-

使用法

この統計を使用すると、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。

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。

エレメント ID

db_work_action_set_id

エレメント・タイプ

情報

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
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。

エレメント ID

db_work_class_id

エレメント・タイプ

情報

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
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。

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。

CoordActLifetime
ネストなしアクティビティーの経過存続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アクティビティーがシステムに入った時からアクティビティーが実行を完了するまでを測定します。存続時間には、アクティビティーが初期化に要する時間、キューに入れられている時間、および実行に要する時間が含まれます。

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。

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。

ReqExecTime
要求の実行時間のヒストグラム。アクティビティーに関連付けられていない要求を含む、コーディネーター・パーティションと非コーディネーター・パーティションの両方におけるすべての要求が含まれます。

エレメント ID
histogram_type

エレメント・タイプ
情報

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
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。

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。

エレメント ID
last_wlm_reset

エレメント・タイプ
情報

表 85. イベント・モニター情報

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 85. 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	-

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。

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。

エレメント ID

num_threshold_violations

エレメント・タイプ

カウンター

表 86. スナップショット・モニター情報

スナップショット・レベル	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 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
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。

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。

エレメント ID

number_in_bin

エレメント・タイプ

情報

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_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。

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。

エレメント ID

parent_activity_id

エレメント・タイプ

情報

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
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。

エレメント ID

parent_uow_id

エレメント・タイプ

情報

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
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。

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)。

エレメント ID

prep_time

エレメント・タイプ

時間

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	-

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。

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割り当てられた接続またはアクティビティーの数。

エレメント ID

queue_assignments_total

エレメント・タイプ

カウンター

表 92. イベント・モニター情報

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この特定のキューに入れられたアクティビティーまたは接続の数を判別できます。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。

エレメント ID

queue_size_top

エレメント・タイプ
水準点

表 93. イベント・モニター情報

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。

エレメント ID
queue_time_total

エレメント・タイプ
カウンター

表 94. イベント・モニター情報

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。

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 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
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_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。

注: このモニター・エレメントは、このレコード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_modified

エレメント・タイプ

カウンター

表 96. イベント・モニター情報

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_written** モニター・エレメントを参照してください。

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。

エレメント ID

rows_returned

エレメント・タイプ

カウンター

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
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。

エレメント ID

rows_returned_top

エレメント・タイプ

水準点

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
統計	event_scstats	-
統計	event_wcstats	-

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。

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。

エレメント ID

sc_work_action_set_id

エレメント・タイプ

情報

表 99. イベント・モニター情報

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。

エレメント ID

sc_work_class_id

エレメント・タイプ

情報

表 100. イベント・モニター情報

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。

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。

エレメント ID

section_env

エレメント・タイプ

情報

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_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。

service_class_id サービス・クラス ID : モニター・エレメント

サービス・クラスのユニーク ID。ヒストグラム・ビン表と結合を行うために使用できます。

エレメント ID

service_class_id

エレメント・タイプ

情報

表 102. イベント・モニター情報

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_scstats	-

使用法

このエレメントを **statistics_timestamp** および **partition_number** モニター・エレメントと一緒に使用すると、ヒストグラム・ビン・レコードをサービス・クラス統計レコードにリンクできます。

service_subclass_name サービス・サブクラス名 : モニター・エレメント

このアクティビティー・レコードまたは統計レコードが適用されるサービス・サブクラスの名前。

エレメント ID

service_subclass_name

エレメント・タイプ

情報

表 103. イベント・モニター情報

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	-
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

このアクティビティー・レコードまたは統計レコードが適用されるサービス・スーパークラスの名前。

エレメント ID

service_superclass_name

エレメント・タイプ

情報

表 104. イベント・モニター情報

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	-
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。

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。

エレメント ID

statistics_timestamp

エレメント・タイプ

情報

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
統計	event_scstats	-
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。

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。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。

エレメント ID

temp_tablespace_top

エレメント・タイプ

水準点

表 106. イベント・モニター情報

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	-

使用法

このエレメントを使用すると、収集された時間間隔にサービス・クラス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。サービス・クラスまたは作業クラスに対する集約アクティビティー・データ収集が有効でない場合、-1 の値が返されます。

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」および「続行」です。

エレメント ID

threshold_action

エレメント・タイプ

情報

表 107. イベント・モニター情報

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、それとも実行を継続でき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、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 エラーを受信したはずです。

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
- ワーク・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

エレメント ID

threshold_domain

エレメント・タイプ

情報

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
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。

エレメント ID

threshold_maxvalue

エレメント・タイプ

情報

表 109. イベント・モニター情報

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。

エレメント ID

threshold_name

エレメント・タイプ

情報

表 110. イベント・モニター情報

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。

エレメント ID

threshold_predicate

エレメント・タイプ

情報

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
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。

エレメント ID

threshold_queuesize

エレメント・タイプ

情報

表 112. イベント・モニター情報

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。

エレメント ID
thresholdid

エレメント・タイプ
情報

表 113. イベント・モニター情報

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 114. イベント・モニター情報

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。

エレメント ID

time_created

エレメント・タイプ

情報

表 115. イベント・モニター情報

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。

エレメント ID

time_of_violation

エレメント・タイプ

情報

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
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_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。

エレメント ID

time_started

エレメント・タイプ
情報

表 117. イベント・モニター情報

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。

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。

エレメント ID
最上位

エレメント・タイプ
情報

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_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。

uow_id 作業単位 ID : モニター・エレメント

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。作業単位 ID は、アプリケーション・ハンドル内で固有です。

エレメント ID
uow_id

エレメント・タイプ
情報

表 119. イベント・モニター情報

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。

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。

エレメント ID
wlo_completed_total

エレメント・タイプ
カウンター

表 120. イベント・モニター情報

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。

エレメント ID
work_action_set_id

エレメント・タイプ
情報

表 121. イベント・モニター情報

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。

エレメント ID

work_action_set_name

エレメント・タイプ

情報

表 122. イベント・モニター情報

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。

エレメント ID

work_class_id

エレメント・タイプ

情報

表 123. イベント・モニター情報

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。

エレメント ID

work_class_name

エレメント・タイプ

情報

表 124. イベント・モニター情報

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

このアクティビティー、アプリケーション、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の ID。

エレメント ID

workload_id

エレメント・タイプ

情報

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
アプリケーション	appl_info	基本

表 126. イベント・モニター情報

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	-
アクティビティー	event_activity	-

使用法

この ID を使用すると、このアクティビティー、アプリケーション、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できます。

workload_name ワークロード名 : モニター・エレメント

この統計レコードが適用されるワークロードの名前。

エレメント ID

workload_name

エレメント・タイプ

情報

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_wlstats	-

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、ワークロードの分析をすることができます。

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。

エレメント ID

workload_occurrence_id

エレメント・タイプ

表 128. イベント・モニター情報

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。

第 10 章 コマンド

SET WORKLOAD コマンド

データベース接続の接続先として割り当てるワークロードを指定します。このコマンドは、データベースに接続する前に発行することができます。あるいは、接続が確立されてから現行接続の再割り当てをするために使用することができます。接続が確立されている場合は、次の作業単位の開始時にワークロードの再割り当てが行われます。

許可

なし

必要な接続

なし

コマンド構文

コマンド・パラメーター

AUTOMATIC
サーバーが自動的に実行するワークロード計算に選ばれているワークロードに、データベース接続を割り当てるよう指定します。

SYSDEFAULTADMWORKLOAD
データベース接続を SYSDEFAULTADMWORKLOAD に割り当てて、*dbadm* 権限または *sysadm* 権限を持つユーザーが通常のワークロード計算を迂回できるよう指定します。

例

接続を SYSDEFAULTADMWORKLOAD に割り当てる方法。

```
SET WORKLOAD TO SYSDEFAULTADMWORKLOAD
```

ワークロード割り当てをリセットして、サーバーが実行するワークロード計算に選ばれているワークロードを使用するようにする方法。

```
SET WORKLOAD TO AUTOMATIC
```

使用上の注意

データベース接続の SESSION 許可 ID に *dbadm* 権限または *sysadm* 権限がない場合、接続を SYSDEFAULTADMWORKLOAD に割り当てることはできず、エラーが戻されます。SET WORKLOAD TO SYSDEFAULTADMWORKLOAD コマンドがデータベース接続前に発行される場合、データベース接続が確立された後の、次の作

© Copyright IBM Corp. 2007, 2008

業単位の開始時にエラーが戻されます。データベース接続が確立されているときにコマンドが発行された場合は、ワークロードの再割り当てが実行される予定の、次の作業単位の開始時にエラーが戻されます。

第 11 章 構成パラメーター

wlm_collect_int - ワークロード管理収集間隔構成パラメーター

このパラメーターは、ワークロード管理 (WLM) 統計の収集およびリセットの間隔を分単位で指定します。

x *wlm_collect_int* 分ごと (x は *wlm_collect_int* パラメーターの値) に、すべてのワークロード管理統計が収集されて、任意のアクティブな統計イベント・モニターに送信され、その後で統計がリセットされます。アクティブなイベント・モニターが存在する場合は、それが作成された方法によって、統計はファイルか表のいずれかに書き込まれます。アクティブなイベント・モニターが存在しない場合は、統計はリセットのみされて、収集はされません。

収集とリセットのプロセスは、カタログ・パーティションから開始されます。カタログ・パーティションでは、*wlm_collect_int* パラメーターを指定する必要があります。これは、他のパーティションでは使用されません。

構成タイプ
データベース

パラメーター・タイプ
オンラインで構成可能

デフォルト [範囲]
0 [0 (収集は実行されない), 5 - 32,767]

統計イベント・モニターにより収集されるワークロード管理統計は、システムの短期および長期の動作をモニターするために使用できます。短い間隔を使用して、システムの短期および長期の動作をモニターすることができます。これは、その結果をマージすると、長期の動作を取得できるためです。ただし、異なる間隔で得られた結果を手動でマージすると、分析が複雑になります。短い間隔の統計が必要なければ、オーバーヘッドが無用に増大するだけです。したがって、長期の動作の分析だけで十分な場合は、短期の動作をキャプチャーする間隔を下げ、オーバーヘッドを削減する間隔を上げます。

この間隔は SQL 要求、コマンドの呼び出し、またはアプリケーションごとではなく、データベースごとにカスタマイズする必要があります。他の構成パラメーターを考慮する必要はありません。

注: すべての WLM 統計表関数は、統計が前回リセットされてから累積した統計を戻します。この統計は、この構成パラメーターで指定された間隔で定期的にリセットされます。

© Copyright IBM Corp. 2007, 2008

第 12 章 カタログ・ビュー

SYSCAT.HISTOGRAMTEMPLATEBINS

各行は、ヒストグラム・テンプレート bin を表します。

表 129. SYSCAT.HISTOGRAMTEMPLATEBINS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TEMPLATENAME	VARCHAR(128)	Y	ヒストグラム・テンプレートの名前。
TEMPLATEID	INTEGER		ヒストグラム・テンプレートの ID。
BINID	INTEGER		ヒストグラム・テンプレート bin の ID。
BINUPPERVALUE	BIGINT		ヒストグラム・テンプレートの 1 つの bin の高い方の値。

SYSCAT.HISTOGRAMTEMPLATES

各行は、ヒストグラム・テンプレートを表します。

表 130. SYSCAT.HISTOGRAMTEMPLATES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TEMPLATEID	INTEGER		ヒストグラム・テンプレートの ID。
TEMPLATENAME	VARCHAR(128)		ヒストグラム・テンプレートの名前。
CREATE_TIME	TIMESTAMP		ヒストグラム・テンプレートが作成された時刻。
ALTER_TIME	TIMESTAMP		ヒストグラム・テンプレートが最後に変更された時刻。
NUMBINS	INTEGER		ヒストグラム・テンプレートの bin の数。これには上限値のない最後の bin も含まれます。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.HISTOGRAMTEMPLATEUSE

各行は、ヒストグラム・テンプレートを使用できるワークロード管理オブジェクトとヒストグラム・テンプレートの関係を表します。

表 131. SYSCAT.HISTOGRAMTEMPLATEUSE カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TEMPLATENAME	VARCHAR(128)	Y	ヒストグラム・テンプレートの名前。

© Copyright IBM Corp. 2007, 2008

表 131. SYSCAT.HISTOGRAMTEMPLATEUSE カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
TEMPLATEID	INTEGER		ヒストグラム・テンプレートの ID。
HISTOGRAMTYPE	CHAR(1)		このテンプレートに基づくヒストグラムで収集される情報のタイプ。 • C = アクティビティー見積コストのヒストグラム • E = アクティビティー実行時間のヒストグラム • I = 1 つのアクティビティーが到着してから別のアクティビティーが到着するまでの時間のヒストグラム • L = アクティビティー存続時間のヒストグラム • Q = アクティビティー・キュー時間のヒストグラム • R = 要求実行時間のヒストグラム
OBJECTTYPE	CHAR(1)		WLM オブジェクトのタイプ。 • b = サービス・クラス • k = 作業アクション
OBJECTID	INTEGER		WLM オブジェクトの ID。
SERVICECLASSNAME	VARCHAR(128)	Y	サービス・クラスの名前。
PARENTSERVICECLASSNAME	VARCHAR(128)	Y	親サービス・クラスの名前。
WORKACTIONNAME	VARCHAR(128)	Y	作業アクションの名前。
WORKACTIONSETNAME	VARCHAR(128)	Y	作業アクション・セットの名前。

SYSCAT.SERVICECLASSES

各行はサービス・クラスを表します。

表 132. SYSCAT.SERVICECLASSES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
SERVICECLASSNAME	VARCHAR(128)		サービス・クラスの名前。
PARENTSERVICECLASSNAME	VARCHAR(128)	Y	親サービス・スーパークラスのサービス・クラス名。
SERVICECLASSID	SMALLINT		サービス・クラスの ID。
PARENTID	SMALLINT		このサービス・クラスの親サービス・クラスの ID。このサービス・クラスがスーパー・サービス・クラスの場合は 0 です。
CREATE_TIME	TIMESTAMP		サービス・クラスが作成された時刻。
ALTER_TIME	TIMESTAMP		サービス・クラスが最後に変更された時刻。

表 132. SYSCAT.SERVICECLASSES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
ENABLED	CHAR(1)		サービス・クラスの状態。 • N = 使用不可 • Y = 使用可能
AGENTPRIORITY	SMALLINT		DB2 スレッドの通常優先順位に対して相対的なサービス・クラス内のエージェントのスレッド優先順位。 • -20 から 20 (Linux および UNIX) • -6 から 6 (Windows®) • -32768 = 設定しない
PREFETCHPRIORITY	CHAR(1)		サービス・クラス内のエージェントのプリフェッチ優先順位。 • H = 高 • L = 低 • M = 中 • ブランク = 設定しない
INBOUNDCORRELATOR	VARCHAR(128)	Y	将来の利用。
OUTBOUNDCORRELATOR	VARCHAR(128)	Y	サービス・クラスとオペレーティング・システムのワークロード・マネージャー・サービス・クラスを関連付けるために使用されるストリング。
COLLECTAGGACTDATA	CHAR(1)		該当するイベント・モニターにサービス・クラスでキャプチャーさせる集約アクティビティー・データを指定します。 • B = 基礎集約アクティビティー・データを収集する • E = 拡張集約アクティビティー・データを収集する • N = なし
COLLECTAGGREQDATA	CHAR(1)		該当するイベント・モニターにサービス・クラスでキャプチャーさせる集約アクティビティー・データを指定します。 • B = 基礎集約要求データを収集する • N = なし
COLLECTACTDATA	CHAR(1)		該当するイベント・モニターによって収集するアクティビティー・データを指定します。 • D = 詳細ありのアクティビティー・データ • N = なし • V = 詳細および値ありのアクティビティー・データ • W = 詳細なしのアクティビティー・データ

表 132. SYSCAT.SERVICECLASSES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
COLLECTACTPARTITION	CHAR(1)		どこでアクティビティー・データを収集するかを指定します。 • C = アクティビティーのコーディネーターのデータベース・パーティション • D = すべてのデータベース・パーティション
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.THRESHOLDS

各行はしきい値を表します。

表 133. SYSCAT.THRESHOLDS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
THRESHOLDNAME	VARCHAR(128)		しきい値の名前。
THRESHOLDID	INTEGER		しきい値の ID。
ORIGIN	CHAR(1)		しきい値の作成元。 • U = しきい値はユーザーによって作成された • W = しきい値は作業アクション・セットから作成された
THRESHOLDCLASS	CHAR(1)		しきい値の分類。 • A = 集約のしきい値 • C = アクティビティーのしきい値
THRESHOLDPREDICATE	VARCHAR(128)		しきい値のタイプ。可能な値は以下のとおりです。 • CONCDBC • CONCWCN • CONCWOC • CONNIDLETIME • DBCONN • ESTSQLCOST • ROWSRET • SCCONN • TEMPSPACE • TOTALTIMECONCDBC
THRESHOLDPREDICATEID	SMALLINT		しきい値の述部の ID。

表 133. SYSCAT.THRESHOLDS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
DOMAIN	CHAR(2)		しきい値のドメイン。 • DB = データベース • SB = サービス・サブクラス • SP = サービス・スーパークラス • WA = 作業アクション・セット • WD = ワークロード定義
DOMAINID	INTEGER		しきい値が関連付けられているオブジェクトの ID。これにはサービス・クラス、作業アクション、またはワークロードの固有 ID を指定できます。これがデータベースのしきい値である場合は、値は 0 です。
ENFORCEMENT	CHAR(1)		しきい値の強制の有効範囲。 • D = データベース • P = データベース・パーティション • W = ワークロード・オカレンス
QUEUEING	CHAR(1)		• N = しきい値はキューイングを行っていない • Y = しきい値はキューイングを行っている
MAXVALUE	BIGINT		しきい値によって指定された上限。
QUEUESIZE	INTEGER		QUEUEING が 'Y' の場合は、キューのサイズ。それ以外の場合は -1。
COLLECTACTDATA	CHAR(1)		該当するイベント・モニターによって収集するアクティビティー・データを指定します。 • A = 詳細なしのアクティビティー・データ • D = 詳細ありのアクティビティー・データ • N = なし • V = 詳細および値ありのアクティビティー・データ
COLLECTACTPARTITION	CHAR(1)		どこでアクティビティー・データを収集するかを指定します。 • C = アクティビティーのコーディネーターのデータベース・パーティション • D = すべてのデータベース・パーティション
EXECUTION	CHAR(1)		しきい値が超過した後、実行を続行するかどうかを指定します。 • C = 実行を続行する • S = 実行を停止する
ENABLED	CHAR(1)		• N = このしきい値は使用不可。 • Y = このしきい値は使用可能。
CREATE_TIME	TIMESTAMP		しきい値が作成された時刻。

表 133. SYSCAT.THRESHOLDS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
ALTER_TIME	TIMESTAMP		しきい値が最後に変更された時刻。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.WORKACTIONS

各行は、作業アクション・セットで定義されている作業アクションを表します。

表 134. SYSCAT.WORKACTIONS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
ACTIONNAME	VARCHAR(128)		作業アクションの名前。
ACTIONID	INTEGER		作業アクションの ID。
ACTIONSETNAME	VARCHAR(128)	Y	作業アクション・セットの名前。
ACTIONSETID	INTEGER		この作業アクションが属する作業アクション・セットの ID。この列は、SYSCAT.WORKACTIONSETS ビューの ACTIONSETID 列を参照します。
WORKCLASSNAME	VARCHAR(128)	Y	作業クラスの名前。
WORKCLASSID	INTEGER		作業クラスの ID。この列は、SYSCAT.WORKCLASSES ビューの WORKCLASSID 列を参照します。
CREATE_TIME	TIMESTAMP		作業アクションが作成された時刻。
ALTER_TIME	TIMESTAMP		作業アクションが最後に変更された時刻。
ENABLED	CHAR(1)		• N = この作業アクションは使用不可です。 • Y = この作業アクションは使用可能です。

表 134. SYSCAT.WORKACTIONS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
ACTIONTYPE	CHAR(1)		作業クラス属性 (突き合わせ有効範囲内の作業クラスで指定された) と一致する各 DB2 アクティビティーで実行されるアクション・タイプ。この列の記述の OBJECTTYPE は、SYSCAT.WORKACTIONSETS の列 OBJECTTYPE を参照します。 • B - 基礎集約アクティビティー・データを収集します。このアクション・タイプは、OBJECTTYPE が 'b' (サービス・クラス) の場合のみ指定できます。 • C - この作業アクションが関連付けられている作業クラスに該当するすべての DB2 アクティビティーの実行を許可し、作業クラスのカウンターを増分します。 • D - アクティビティーのコーディネーターのデータベース・パーティションにおける、詳細を含むアクティビティー・データを収集します。 • E - 拡張された集約アクティビティー・データを収集します。このアクション・タイプは、OBJECTTYPE が 'b' (サービス・クラス) の場合のみ指定できます。 • M - サービス・サブクラスにマップします。このアクション・タイプは、OBJECTTYPE が 'b' (サービス・クラス) の場合のみ指定できます。 • P - この作業アクションが関連付けられている作業クラスに該当する DB2 アクティビティーが実行されないようにします。 • T - アクションはしきい値の形式になります。このアクション・タイプは、OBJECTTYPE が 'f' (しきい値) の場合のみ指定できます。 • U - ネスト・レベルがゼロのアクティビティーおよびこのアクティビティーの下にネストされているアクティビティーすべてをサービス・サブクラスにマップします。このアクション・タイプは、OBJECTTYPE が 'b' (サービス・クラス) の場合のみ指定できます。

表 134. SYSCAT.WORKACTIONS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
ACTIONTYPE (cont'd)			• V - アクティビティーのコーディネーターのデータベース・パーティションにおける、詳細と値を含むアクティビティー・データを収集します。 • W - アクティビティーのコーディネーターのデータベース・パーティションにおける、詳細を含まないアクティビティー・データを収集します。 • X - アクティビティーのコーディネーターのデータベース・パーティションにおける、詳細を含むアクティビティー・データと、すべてのデータベース・パーティションにおけるアクティビティー・データを収集します。 • Y - アクティビティーのコーディネーターのデータベース・パーティションにおける、詳細と値を含むアクティビティー・データと、すべてのデータベース・パーティションにおけるアクティビティー・データを収集します。 • Z - すべてのデータベース・パーティションにおける、詳細を含まないアクティビティー・データを収集します。
REFOBJECTID	INTEGER	Y	ACTIONTYPE が 'M' (マップ) または 'N' (ネストされたマップ) の場合、この値は DB2 アクティビティーのマップ先のサービス・サブクラスの ID に設定されます。ACTIONTYPE が 'T' (しきい値) の場合、この値は、使用されるしきい値の ID に設定されます。それ以外のすべてのアクションの場合、この値は NULL になります。
REFOBJECTTYPE	VARCHAR(30)		ACTIONTYPE が 'M' または 'N' の場合、この値は 'SERVICE CLASS' に設定されます。ACTIONTYPE が 'T' の場合、この値は 'THRESHOLD' になり、それ以外の場合は NULL 値になります。

SYSCAT.WORKACTIONSETS

各行は、作業アクション・セットを表します。

表 135. SYSCAT.WORKACTIONSETS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
ACTIONSETNAME	VARCHAR(128)		作業アクション・セットの名前。

表 135. SYSCAT.WORKACTIONSETS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
ACTIONSETID	INTEGER		作業アクション・セットの ID。
WORKCLASSSETNAME	VARCHAR(128)	Y	作業クラス・セットの名前。
WORKCLASSSETID	INTEGER		OBJECTID で指定されたオブジェクトにマップされる作業クラス・セットの ID。この列は SYSCAT.WORKCLASSSETS ビューの WORKCLASSSETID を参照します。
CREATE_TIME	TIMESTAMP		作業アクション・セットが作成された時刻。
ALTER_TIME	TIMESTAMP		作業アクション・セットが最後に変更された時刻。
ENABLED	CHAR(1)		• N = この作業アクション・セットは使用不可です。 • Y = この作業アクション・セットは使用可能です。
OBJECTTYPE	CHAR(1)		• b = サービス・スーパークラス • ブランク = データベース
OBJECTNAME	VARCHAR(128)	Y	サービス・クラスの名前。
OBJECTID	INTEGER		作業クラス・セット (WORKCLASSSETID で指定される) のマップ先のオブジェクトの ID。OBJECTTYPE がブランクの場合、OBJECTID は -1 です。 OBJECTTYPE が 'b' の場合、OBJECTID はサービス・スーパークラスの ID です。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.WORKCLASSES

各行は、作業クラス・セットで定義されている作業クラスを表します。

表 136. SYSCAT.WORKCLASSES カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
WORKCLASSNAME	VARCHAR(128)		作業クラスの名前。
WORKCLASSSETNAME	VARCHAR(128)	Y	作業クラス・セットの名前。
WORKCLASSID	INTEGER		作業クラスの ID。
WORKCLASSSETID	INTEGER		この作業クラスが属する作業クラス・セットの ID。この列は、SYSCAT.WORKCLASSSETS ビューの WORKCLASSSETID 列を参照します。
CREATE_TIME	TIMESTAMP		作業クラスが作成された時刻。
ALTER_TIME	TIMESTAMP		作業クラスが最後に変更された時刻。

表 136. SYSCAT.WORKCLASSES カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
WORKTYPE	SMALLINT		DB2 アクティビティーのタイプ。 • 1 = ALL • 2 = READ • 3 = WRITE • 4 = CALL • 5 = DML • 6 = DDL • 7 = LOAD
RANGEUNITS	CHAR(1)		上下範囲に使用する単位。 • C = カーディナリティー • T = Timeron • ブランク = 該当しない場合
FROMVALUE	DOUBLE	Y	RANGEUNITS で指定された単位による、範囲の低い値。 RANGEUNITS がブランクの場合は NULL 値になります。
TOVALUE	DOUBLE	Y	RANGEUNITS で指定された単位による、範囲の高い値。 RANGEUNITS がブランクの場合は NULL 値になります。値 -1 は上限なしを示すために使われます。
ROUTINESCHEMA	VARCHAR(128)	Y	CALL ステートメントから呼び出されるプロシージャーのスキーマ名。 WORKTYPE が 4 (CALL) または 1 (ALL) でない場合は NULL 値。
EVALUATIONORDER	SMALLINT		作業クラス・セット内の作業クラスを選択するときに使用される評価順序を一意的に識別します。

SYSCAT.WORKCLASSSETS

各行は、作業クラス・セットを表します。

表 137. SYSCAT.WORKCLASSSETS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
WORKCLASSSETNAME	VARCHAR(128)		作業クラス・セットの名前。
WORKCLASSSETID	INTEGER		作業クラス・セットの ID。
CREATE_TIME	TIMESTAMP		作業クラス・セットが作成された時刻。
ALTER_TIME	TIMESTAMP		作業クラス・セットが最後に変更された時刻。
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

SYSCAT.WORKLOADAUTH

各行は、ワークロードに対する USAGE 特権が付与されているユーザー、グループ、またはロールを表します。

表 138. SYSCAT.WORKLOADAUTH カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
WORKLOADID	INTEGER		ワークロードの ID。
WORKLOADNAME	VARCHAR(128)		ワークロードの名前。
GRANTOR	VARCHAR(128)		特権の認可者。
GRANTORTYPE	CHAR(1)		• U = GRANTEE は個々のユーザー。
GRANTEE	VARCHAR(128)		特権の保有者。
GRANTEETYPE	CHAR(1)		• G = GRANTEE はグループ。 • R = GRANTEE はロール。 • U = GRANTEE は個々のユーザー。
USAGEAUTH	CHAR(1)		GRANTEE がワークロードに対する USAGE 特権を保有するかどうかを示します。 • N = 保有しない • Y = 保有する

SYSCAT.WORKLOADCONNATTR

各行は、ワークロードの定義における接続属性を表します。

表 139. SYSCAT.WORKLOADCONNATTR カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
WORKLOADID	INTEGER		ワークロードの ID。
WORKLOADNAME	VARCHAR(128)		ワークロードの名前。
CONNATTRTYPE	VARCHAR(30)		接続属性のタイプ。 • 1 = APPLNAME • 2 = SYSTEM_USER • 3 = SESSION_USER • 4 = SESSION_USER GROUP • 5 = SESSION_USER ROLE • 6 = CURRENT CLIENT_USERID • 7 = CURRENT CLIENT_APPLNAME • 8 = CURRENT CLIENT_WRKSTNNAME • 9 = CURRENT CLIENT_ACCTNG
CONNATTRVALUE	VARCHAR(1000)		接続属性の値。

SYSCAT.WORKLOADS

各行はワークロードを表します。

表 140. SYSCAT.WORKLOADS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
WORKLOADID	INTEGER		ワークロードの ID。
WORKLOADNAME	VARCHAR(128)		ワークロードの名前。
EVALUATIONORDER	SMALLINT		ワークロードの選択に使用される評価順序。
CREATE_TIME	TIMESTAMP		ワークロードが作成された時刻。
ALTER_TIME	TIMESTAMP		ワークロードが最後に変更された時刻。
ENABLED	CHAR(1)		• N = このワークロードは使用不可。 • Y = このワークロードは使用可能。
ALLOWACCESS	CHAR(1)		• N = このワークロードに関連付けられている UOW は拒否される。 • Y = このワークロードに関連付けられている作業単位 (UOW) はデータベースにアクセスできる。
SERVICECLASSNAME	VARCHAR(128)		(このワークロードに関連付けられている) 作業単位が割り当てられるサービス・サブクラスの名前。
PARENTSERVICECLASSNAME	VARCHAR(128)	Y	(このワークロードに関連付けられている) 作業単位が割り当てられるサービス・スーパークラスの名前。
COLLECTAGGACTDATA	CHAR(1)		このワークロードについて、該当するイベント・モニターでキャプチャーする集約アクティビティー・データを指定します。 • N = なし
COLLECTACTDATA	CHAR(1)		該当するイベント・モニターによって収集するアクティビティー・データを指定します。 • D = 詳細ありのアクティビティー・データ • N = なし • V = 詳細および値ありのアクティビティー・データ (COLLECT 列が 'C' に設定されている場合に適用される) • W = 詳細なしのアクティビティー・データ
COLLECTACTPARTITION	CHAR(1)		どこでアクティビティー・データを収集するかを指定します。 • C = アクティビティーのコーディネーターのデータベース・パーティション • D = すべてのデータベース・パーティション
EXTERNALNAME	VARCHAR(128)	Y	将来の使用のために予約されています。

表 140. SYSCAT.WORKLOADS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
REMARKS	VARCHAR(254)	Y	ユーザー提供のコメントまたは NULL 値。

第 6 部 付録

© Copyright IBM Corp. 2007, 2008

付録 A. ワークロード管理 DDL ステートメントの考慮事項

DB2 ワークロード管理 DDL ステートメントは CREATE、ALTER、および DROP の各ステートメントから構成されます。これらは、サービス・クラス、ワークロード、ワーク・クラス・セット、ワーク・アクション・セット、しきい値、およびヒストグラムを操作するときに使用します。

ワークロード管理 DDL ステートメントを以下に示します。

- CREATE WORKLOAD、ALTER WORKLOAD、および DROP WORKLOAD
- GRANT USAGE ON WORKLOAD および REVOKE USAGE ON WORKLOAD
- CREATE SERVICE CLASS、ALTER SERVICE CLASS、および DROP SERVICE CLASS
- CREATE WORK CLASS SET、ALTER WORK CLASS SET、および DROP WORK CLASS SET
- CREATE WORK ACTION SET、ALTER WORK ACTION SET、および DROP WORK ACTION SET
- CREATE THRESHOLD、ALTER THRESHOLD、および DROP THRESHOLD
- CREATE HISTOGRAM TEMPLATE、ALTER HISTOGRAM TEMPLATE、および DROP HISTOGRAM TEMPLATE

ワークロード管理 DDL ステートメントは、以下のように他の DB2 DDL ステートメントとは異なります。

- 全データベース・パーティションにおいて、非コミット・ワークロード管理 DDL ステートメントは、一度に 1 つしか許可されません。非コミット・ワークロード管理 DDL ステートメントが存在する場合、それ以降のワークロード管理 DDL ステートメントは非コミット・ワークロード管理 DDL ステートメントがコミットされるかロールバックされるまで待機します。ワークロード管理 DDL ステートメントは、発行された順序に処理されます。
- すべてのワークロード管理 DDL ステートメントの後は、COMMIT または ROLLBACK ステートメントでなければなりません。
- ワークロード管理 DDL ステートメントは、XA トランザクションでは発行できません。ある接続でワークロード管理 DDL ステートメントを発行した後、同じ接続でそのワークロード管理 DDL ステートメントの直後に COMMIT または ROLLBACK ステートメントを発行する必要があります。 XA トランザクションでは、複数の接続がトランザクションに参加することが可能で、いずれの接続もトランザクションをコミットまたはロールバックできます。この状況では、ワークロード管理環境を正常にインプリメントすることは不可能です。
- DB2 for z/OS® は、DB2 Database for Linux, UNIX, and Windows のワークロード管理 DDL ステートメントを認識しません。

多くのデータベース・オブジェクトには所有者がいて、これらの所有者には自分が所有するオブジェクトを変更する権限があります。ほとんどのオブジェクトとは異なり、ワークロード管理オブジェクトには所有者がいません。所有者がいると、予測不能な問題が発生するためです。例えば、あるサービス・クラスのリソース割り

© IBM Corporation 2007, 2008

振り設定が変更されると、その変更の影響はそのサービス・クラスそのものだけでなく、同じ層の他のサービス・クラスにも及びます。例えば、あるサービス・スーパークラスに 2 つのユーザー定義のサービス・サブクラス、A および B があり、各サービス・サブクラスの所有者がそれぞれ異なると仮定します。デフォルトのサービス・サブクラスおよび 2 つのサービス・サブクラス A および B のプリフェッチ優先順位設定は、中です。サービス・サブクラス A の所有者がそのプリフェッチ優先順位を高に変更し、このサービス・サブクラスから多数のプリフェッチ要求が来た場合、サービス・サブクラス B およびデフォルトのサブクラスへの接続ではプリフェッチャー・サービスが*利用できない状態* になり、これらのサービス・サブクラスで実行しているアクティビティーのパフォーマンスが低下する可能性があります。このような理由で、DB2 ワークロード管理オブジェクトには所有者がありません。

付録 B. DB2 ワークロード管理と AIX ワークロード・マネージャーの統合

DB2 ワークロード管理と AIX ワークロード・マネージャーとを併用する場合は、追加の機能が使用可能です。 AIX ワークロード・マネージャーは、各サービス・クラスに割り振られる CPU の量を制御するために使用できます。

オプションには、各サービス・クラスに対する CPU の最小、最大、または相対比率の共有の設定が含まれます。 DB2 サービス・クラスと AIX ワークロード・マネージャーのサービス・クラスとの間のマッピングは、CREATE SERVICE CLASS または ALTER SERVICE CLASS ステートメントの OUTBOUND CORRELATOR オプションを使用して、DB2 サービス・クラスの定義内に指定できます。 DB2 サービス・クラスは、DB2 ワークロード管理と AIX ワークロード・マネージャーとの統合を図るための唯一の地点となります。

このトピックでは以下を説明しています。

- DB2 サービス・クラスと AIX ワークロード・マネージャーのサービス・クラスとの間の推奨マッピング
- DB2 サービス・クラスと AIX ワークロード・マネージャーのサービス・クラスとの間のマッピングの定義のための手順
- AIX ワークロード・マネージャーのサービス・クラスに対する CPU 制御の設定

DB2 サービス・クラスと AIX ワークロード・マネージャーのサービス・クラスとの間の推奨マッピング

任意の方法で DB2 サービス・クラスを AIX ワークロード・マネージャーのサービス・クラスにマップすることができます。ただし、DB2 サービス・クラスと AIX ワークロード・マネージャーのサービス・クラスの 1:1 マッピングから始めてみる必要があります。 DB2 サービス・クラスのアクティビティーのパフォーマンス目標はほぼ同じであるため、1:1 マッピングは開始点として適切です。その目標を達成する最も直接的な方法は、その AIX ワークロード・マネージャーのサービス・クラスに割り振られる AIX リソースを調整することです。 DB2 サービス・クラスと AIX ワークロード・マネージャーのサービス・クラスとの間で 1:1 マッピングを行うことによって、各サービス・クラスの AIX リソースを個別に調整することができます。

以下の図は、DB2 ワークロード・マネージャーと AIX ワークロード・マネージャーの統合を示しています。サービス・スーパークラスとサービス・サブクラスのレベルでの、DB2 サービス・クラスと AIX ワークロード・マネージャーのサービス・クラスとの間での 1:1 マッピングに注目してください。

© IBM Corporation 2007, 2008

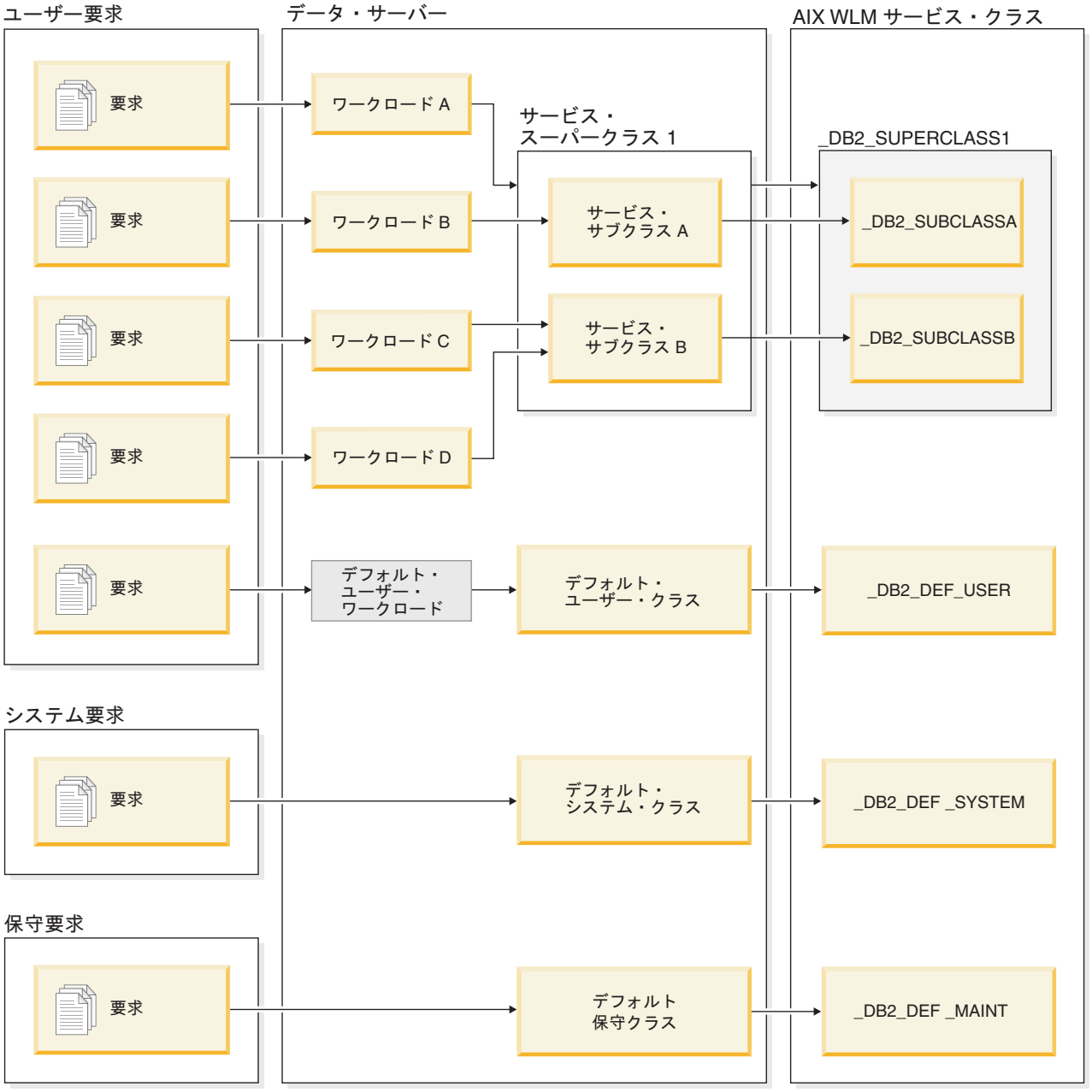


図 28. DB2 ワークロード・マネージャーと AIX ワークロード・マネージャーの統合

前の図で示す例のように、DB2 環境が単一の DB2 インスタンス内の単一のデータベースで構成される場合、DB2 サービス・クラスと AIX ワークロード・マネージャーのサービス・クラスとの間での直接のマップが可能です。それぞれの DB2 サービス・スーパークラスは、対応する AIX ワークロード・マネージャーのサービス・スーパークラスを持つことができ、それぞれの DB2 サービス・サブクラスは、対応する AIX サービス・サブクラスにマップすることができます。

以下の図は、DB2 環境が複数のデータベースと DB2 インスタンスとで構成されている状況で、4 つのレベルがリソース制御の候補であることを示しています。

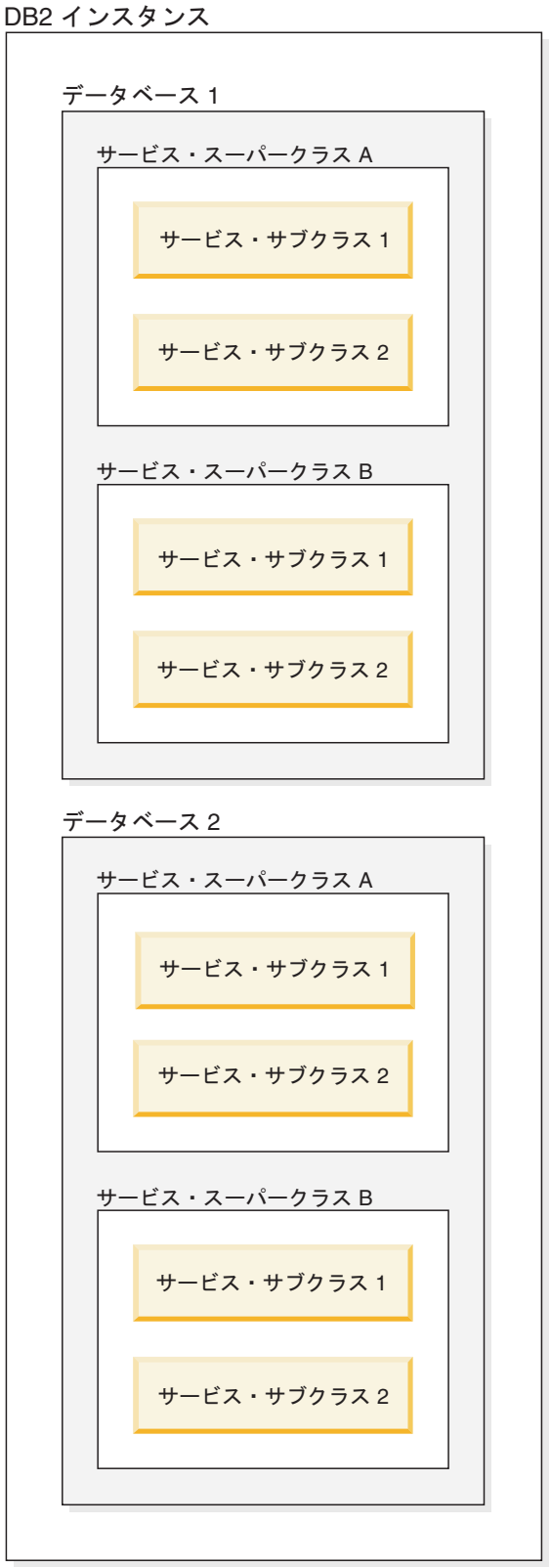


図 29. DB2 環境におけるリソース制御レベル

AIX ワークロード・マネージャーはスーパークラスとサブクラスの 2 つのレベルの階層をサポートしているので、DB2 環境の 2 つのレベルだけは AIX ワークロード・マネージャーのサービス・クラスにいつでもマップできます。いくつかの構成例を以下に示します。

以下の図は、それぞれスーパークラスを持つ複数のデータベースがある場合に、1:1 マッピングを実現するための 1 つの方法を示しています。ここで、各データベースにはその固有の AIX ワークロード・マネージャーのスーパークラスがあり、各 DB2 サービス・スーパークラスは AIX ワークロード・マネージャーのサブクラスにマップされています。

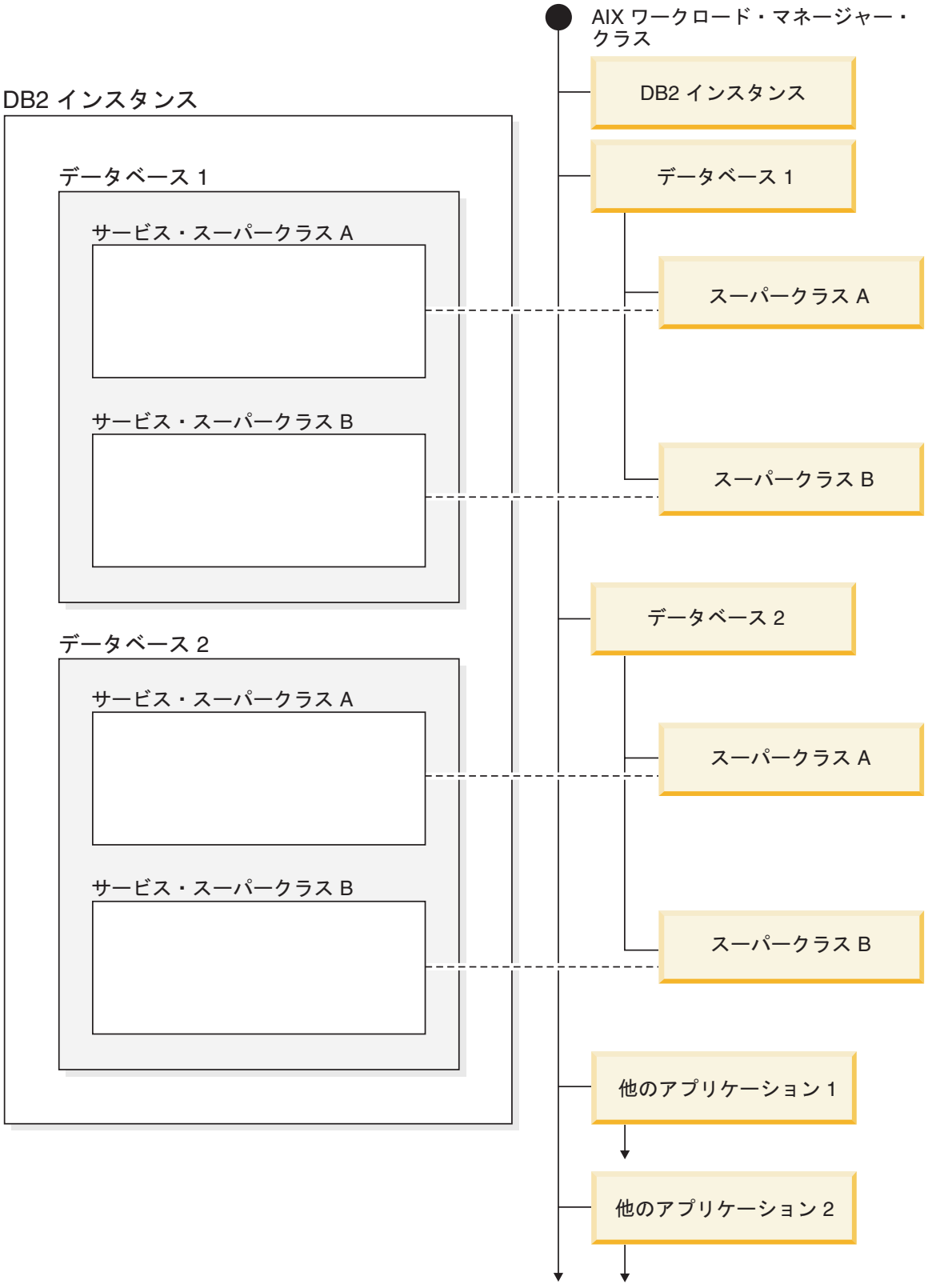


図 30. AIX サービス・クラスにマップされる DB2 サービス・クラス (DB2 サービス・スーパークラスのみを含む)

代替構成として、各 DB2 サービス・スーパークラスをその固有の AIX ワークロード・マネージャーのスーパークラスにマップすることができます。この例では結果

として 4 つのスーパークラスになります。この状況では、データベース・レベルのリソース制御は、AIX ワークロード・マネージャーのサービス・クラス定義で明示的に表されます。

以下の図は、それぞれサービス・スーパークラスとサービス・サブクラスを持つ複数のデータベースがある場合に、1:1 マッピングを実現するための 1 つの方法を示しています。ここで、各データベースは AIX スーパークラスにマップされ、各 DB2 サービス・サブクラスは AIX ワークロード・マネージャーのサブクラスにマップされます。 DB2 サービス・スーパークラスは、AIX ワークロード・マネージャーのサービス・クラス定義には明示的に示されません。

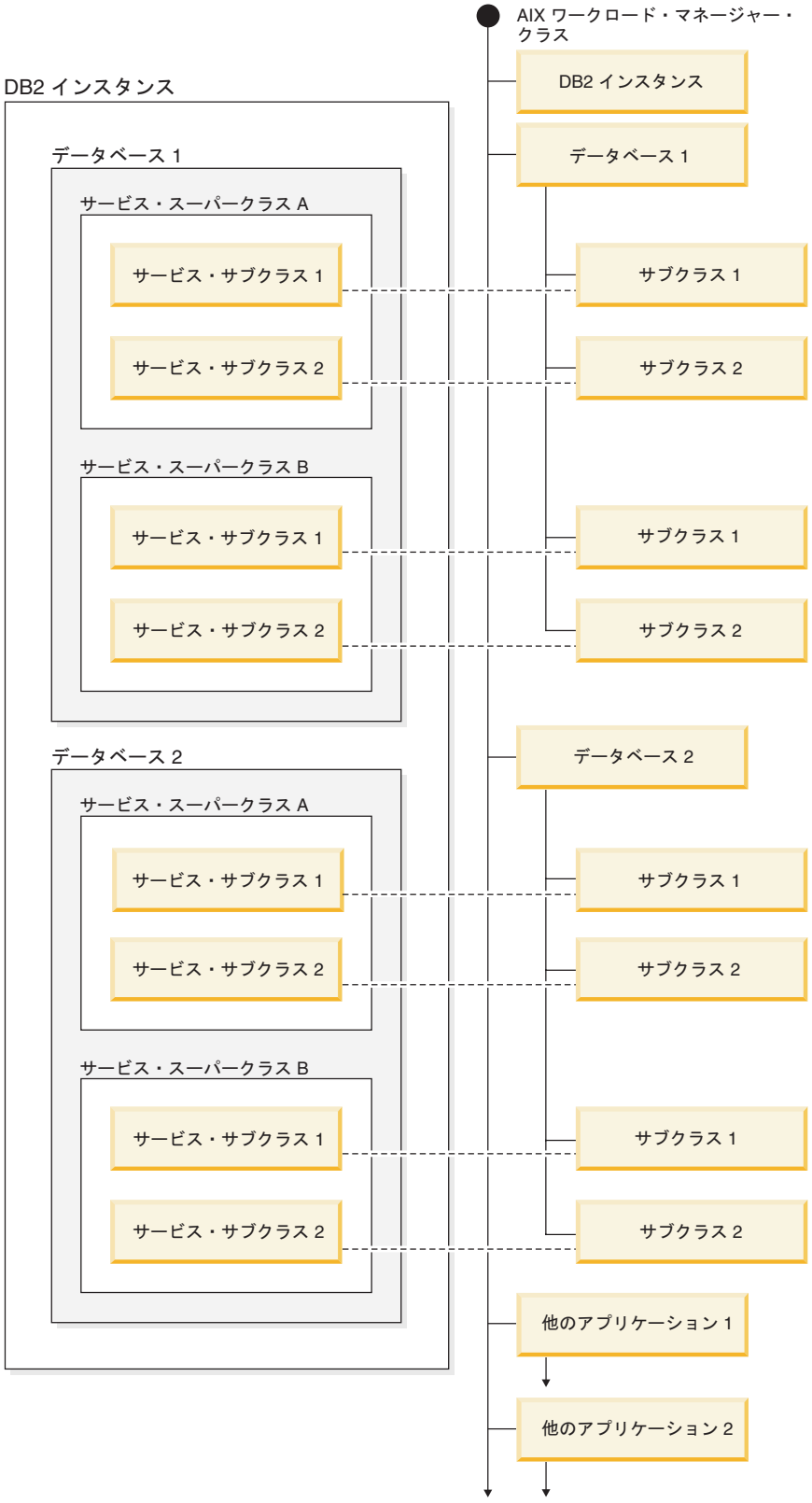


図 31. AIX ワークロード・マネージャーのクラスにマップされる DB2 サービス・クラス (DB2 サービス・サブクラスを含む)

DB2 サービス・クラスと AIX ワークロード・マネージャーのサービス・クラスとの間のマッピングの定義

DB2 サービス・クラスと AIX ワークロード・マネージャーのサービス・クラスとの間のマッピングは、CREATE SERVICE CLASS または ALTER SERVICE CLASS ステートメントの OUTBOUND CORRELATOR キーワードを使用して、DB2 サービス・クラスに指定されます。

DB2 データ・サーバーを使用して AIX ワークロード・マネージャーのサービス・クラスをセットアップする手順は以下のとおりです。

1. DB2 サービス・スーパークラスとサービス・サブクラスを作成し、OUTBOUND CORRELATOR タグを指定します。
2. それに対応する AIX サービス・クラスを作成します。
3. DB2 ワークロード管理と AIX ワークロード・マネージャーとのマッピングを含めるための、関連した AIX ワークロード・マネージャー規則ファイルを、タグ列の下で OUTBOUND CORRELATOR タグを使用して作成します。
4. AIX ワークロード・マネージャーを開始します。
5. 必要な場合、この AIX ワークロード・マネージャー構成をアクティブに設定します。

以下のポイントでは、DB2 バージョン 9.5 の UNIX および Linux 上のプロセス・モデルからスレッド化モデルへの変更がある場合の、AIX ワークロード・マネージャーによる DB2 データ・サーバーからの作業の処理方法を説明しています。スレッドと DB2 サービス・クラスが結合されると、DB2 データ・サーバーは該当する AIX ワークロード・マネージャー API を呼び出して、そのスレッドを対応する AIX サービス・クラスと関連付けます。 DB2 データ・サーバーは、OUTBOUND CORRELATOR パラメーターで設定されているアプリケーション・タグを渡すことによって、そのスレッドのターゲット AIX サービス・クラスを AIX ワークロード・マネージャーに送信します。

AIX ワークロード・マネージャーが正しくインストールおよび構成されており、アクティブになっていることを確認する必要があります。 DB2 データ・サーバーが AIX ワークロード・マネージャーと通信できない場合、メッセージが db2diag.log および DB2 管理者ログに記録されます。データベース・アクティビティーは続行します。

DB2 データ・サーバーは、AIX ワークロード・マネージャーに渡す OUTBOUND CORRELATOR 値が AIX ワークロード・マネージャーによって認識されているかどうかを検出できません。 DB2 サービス・クラスに指定されている値が、DB2 スレッドを AIX サービス・クラスにマップするアプリケーション・タグと一致していることを確認する必要があります。 OUTBOUND CORRELATOR 値が AIX ワークロード・マネージャーによって認識されていない場合、データベース・アクティビティーは実行を続行します。

他の注目すべき点には以下があります。

- DB2 サービス・クラスは、AIX ワークロード・マネージャーの継承フィーチャーを処理することができません。継承は AIX サービス・クラスのデフォルト設定です。継承は継承属性を NO に設定することで明示的に使用不可にする必要があ

ります。AIX ワークロード・マネージャーの継承によって、すべての子スレッドおよびプロセスは親スレッドまたはプロセスと同じクラスに強制的にマップされます。継承が使用可能な場合、DB2 ワークロード・マネージャーは、タグを使用してスレッドの AIX ワークロード管理クラスを変更することはできません。この制限により、DB2 ワークロード・マネージャーと AIX ワークロード・マネージャーの一切の統合は使用できなくなります。 DB2 データ・サーバーは、AIX ワークロード・マネージャーの継承が使用可能かどうかを検出できないため、継承が使用可能な場合でもエラー・メッセージは発行されません。

- DB2 サービス・クラスには、AIX ワークロード・マネージャーの手動割り当てフィーチャーとの互換性はありません。手動割り当てフィーチャーを使用した場合、ユーザーは特定の AIX ワークロード・マネージャーのクラスに処理を手動で割り当てることができます。 DB2 プロセスを手動で割り当てることにより、プロセス内のすべてのスレッドはターゲット AIX ワークロード・マネージャーのクラスに割り当てられ、DB2 サービス・クラスのマッピング・ロジックは無効になり、結果は予測できません。
- AIX 5.3.H 以降が必要です。

AIX ワークロード・マネージャーについて詳しくは、http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp の AIX インフォメーション・センターを参照してください。

AIX ワークロード・マネージャーのサービス・クラスに対する CPU 制御の設定

AIX ワークロード・マネージャーは、各サービス・クラスに割り振られる CPU の量を制御するために使用できます。オプションには、各サービス・クラスに対する CPU の最小、最大、または相対比率の共有の設定が含まれます。

AIX ワークロード・マネージャーと DB2 ワークロード管理を統合する場合、CPU 割り振り制御のみがサポートされます。 AIX サービス・クラスのメモリーおよび入出力設定を行わないでください。 DB2 データベース・レベルのメモリーは、異なる DB2 サービス・クラスのすべてのエージェントで共有されるため、異なるサービス・クラス間でメモリー割り振りを分割することはできません。 AIX レベルの入出力制御は、新しい DB2 エンジン・スレッド化モデルをサポートしません。入出力を制御するには、DB2 サービス・クラスのプリフェッチャー優先順位属性を使用して、異なる DB2 サービス・クラス間での入出力の優先順位を区別することができます。

AIX を使用してサービス・クラスに割り振られる CPU の量を制御する場合は、その DB2 サービス・クラスのエージェント優先順位設定も変更しないでください。CPU リソースへのアクセスを管理するには、これらのメカニズムの 1 つだけを使用します。サービス・クラスに AGENT PRIORITY および OUTBOUND CORRELATOR 値の両方を設定することはできません。詳しくは、33 ページの『CPU 優先順位および DB2 サービス・クラス』を参照してください。

AIX ワークロード・マネージャーの設定は、インスタンスに関与するすべての物理コンピューターで一貫していなければなりません。例えば、あるコンピューターで AIX サービス・クラスのリソース設定が高く設定されている場合、他のすべてのコンピューターのその AIX サービス・クラスに同じ設定を使用しなければなりませ

ん。リソース使用量の設定がコンピューター間で一貫していない場合、同じ AIX サービス・クラスで実行されている要求は、別のデータベース・パーティション上で異なるパフォーマンス・レベルを示します。この状態によって、AIX サービス・クラスにおける接続の全体のスループットが落ちる可能性があります。

付録 C. ワークロード管理ソリューションにおけるストアード・プロシージャーの処理

ストアード・プロシージャーの各呼び出しは 1 つのアクティビティーとして処理されます。ストアード・プロシージャーによって発行されるすべてのデータベース・アクティビティーは、ストアード・プロシージャーの子アクティビティーとして処理されます。マッピング・ワーク・アクションがストアード・プロシージャーに適用されると、その構成に応じて、ストアード・プロシージャーの子アクティビティーは、親アクティビティーと同じサービス・サブクラスまたは別のサービス・サブクラスで実行することができます。

アクティビティーの並行性のしきい値は、ストアード・プロシージャー自体とその子アクティビティーに適用されます。ストアード・プロシージャーの実行がキューに入れられると、その子アクティビティーはどれも続行できなくなります。ただし、ストアード・プロシージャーが実行を開始するときに、子アクティビティーがキューに入れられる場合があります。

fenced モードで実行されるストアード・プロシージャーの場合、WLM_GET_SERVICE_CLASS_AGENTS 表関数を使用して、ストアード・プロシージャーのプロセスのプロセス ID またはスレッドのスレッド ID を表示することができます。

© IBM Corporation 2007, 2008

付録 D. 命名規則

すべてのオブジェクト、ユーザー、およびグループの命名には規則があります これらの規則には、作業しているプラットフォームに特有のものもあります。

たとえば、名前に大文字と小文字を使用することに関連した規則があります。

- UNIX プラットフォームでは、名前は小文字でなければなりません。
- Windows プラットフォームでは、名前は大文字でも、小文字でも、大/小文字混合でも構いません。

特に指定がない限り、名前には以下の文字を含めることができます。

- A から Z までの文字。名前に使用されるとき、多くの場合 A から Z は小文字から大文字に変換されます。
- 0 から 9 の数字。
- ! % ( ) { } . – ^ ~ _ (アンダースコアー) @、#、$、およびスペース。
- ¥ (円記号)。

名前の先頭に、数値または下線文字を使用することはできません。

表、ビュー、列、索引、または許可 ID の名前には、SQL 予約語を使用しないでください。

ご使用のオペレーティング・システム、および DB2 データベースを操作している場所によって、異なる働きをする特殊文字が他にもあります。 それらの文字は正常に機能する可能性がありますが、必ず機能するという保証はありません。データベース内のオブジェクトを命名する際には、これら他の特殊文字を使用することはお奨めしません。

ユーザーおよびグループ名は、関連したシステムによって特定のオペレーション・システムに強制された規則にも従う必要があります。例えば、Linux および UNIX プラットフォームでは、ユーザー名および 1 次グループ名に許可されている文字は、名前として、a から z までの小文字、0 から 9 までの数字、および _ (下線) である必要があります (ただし、0 から 9 までの数字で始まらないこと)。

各長さは、SQL および XML の制限値にリストされた長さ以下である必要があります。

他にも、オブジェクト命名規則、NLS 環境での命名規則、および Unicode 環境での命名規則を考慮する必要があります。

AUTHID ID に関する制約事項: DB2 データベース・システムのバージョン 9.5 以降では、128 バイトの許可 ID を付けることが可能ですが、許可 ID がオペレーティング・システムのユーザー ID またはグループ名として解釈される場合は、オペレーティング・システムの命名上の制約事項が適用されます (例えば、8 または 30 文字のユーザー ID と 30 文字のグループ名の制限)。このため、128 バイトの許可 ID を付与できますが、この許可 ID を持つユーザーとしては接続することができません。独自のセキュリティー・プラグインを作成した場合は、許可 ID の拡張され

© IBM Corporation 2007, 2008

たサイズを最大限に活用することができます。例えば、セキュリティー・プラグインに 30 バイトのユーザー ID を与えて、接続可能な認証中に、セキュリティー・プラグインが 128 バイトの許可 ID を返すことができます。

付録 E. ロール

ロールは、特権の管理を簡素化します。つまり、グループと同等の機能は提供されますが、同じ制約事項は設けられません。ロールは、1 つ以上の特権をまとめたデータベース・オブジェクトですが、これを、GRANT ステートメントを使用してユーザー、グループ、PUBLIC、またはその他のロールに割り当てるか、あるいは、CREATE TRUSTED CONTEXT または ALTER TRUSTED CONTEXT ステートメントを使用して、トラステッド・コンテキストに割り当てることができます。ワークロード定義内で、SESSION_USER ROLE 接続属性用のロールを指定することができます。

ロールは、次のように、データベース・システム内での特権の管理がより簡単になるという利点を備えています。

- セキュリティー管理者は、組織の構造を映し出すような方法で、そのデータベースへのアクセスを制御することができます (組織における役職や担当業務に対応したロールをデータベース内に作成できます)。
- ユーザーには、その役職や担当業務に応じたロールに対するメンバーシップが付与されます。ユーザーの役職や担当業務の変更に応じて、ロールに対するメンバーシップを簡単に付与または取り消すことができます。
- 特権の割り当てが簡素化されます。管理者は、特定の役職や担当業務に該当する個々のユーザーに一連の同じ特権を付与するのではなく、その役職や担当業務に応じたロールに対してこの一連の特権を付与してから、その役職や担当業務に該当する各ユーザーにそのロールを付与することができます。
- そのロールを付与されたすべてのユーザーに対し、更新が適用されます。つまり管理者は、個人ごとに各ユーザーの特権を更新する必要はありません。
- ビュー、トリガー、マテリアライズ照会表 (MQT)、静的 SQL、および SQL ルーチンの作成時には、ロールに対して付与された特権および権限が常に使用されます。この場合、グループに付与された特権および権限は (直接でも間接にでも) 使用されません。

  その理由は、グループはサード・パーティー・ソフトウェア (例えば、オペレーティング・システムまたは LDAP ディレクトリー) によって管理されるので、DB2 データベース・システムは、グループ内のメンバーシップがいつ変更になったかを判別できないからです。ロールはデータベース内部で管理されるので、DB2 データベース・システムは、許可がいつ変更されたかを判別して、それに応じたアクションをとることができます。グループが検討の対象にならないのと同じ理由で、グループに付与されたロールも検討の対象にはなりません。
- ユーザーに割り当てられたすべてのロールは、ユーザーが接続を確立したときに有効になるので、ロールに付与されたすべての特権と許可も、ユーザーが接続するときに有効となります。ロールを明示的に有効または無効にすることはできません。
- セキュリティー管理者は、ロールの管理を他人に委任することができます。

© IBM Corporation 2007, 2008

セキュリティー管理者 (SECADM) 権限を例外として、データベース内で付与できるどの DB2 特権および権限でも、ロールに付与することができます。例えば、以下のどの権限および特権でも、ロールに付与することができます。

- DBADM、LOAD、および IMPLICIT_SCHEMA データベース権限
- CONNECT、CREATETAB、CREATE_NOT_FENCED、BINDADD CREATE_EXTERNAL_ROUTINE、または QUIESCE_CONNECT データベース権限
- 任意のデータベース・オブジェクト特権 (CONTROL を含む)

ユーザーがデータベースに接続したとき、そのユーザーのロールは自動的に有効になり、許可の検討の対象になります。つまり、SET ROLE ステートメントを使用してロールを活動化する必要はありません。例えば、ビュー、マテリアライズ照会表 (MQT)、トリガー、パッケージ、または SQL ルーチンを作成すると、ロールを通して取得した特権が適用されます。ただし、自分がメンバーとして所属するグループに付与されたロールを通して取得した特権は適用されません。

ロールには所有者はいません。セキュリティー管理者は、GRANT ステートメントの WITH ADMIN OPTION 節を使用して、ロールの管理を別のユーザーに委任することができます。それによって、他のユーザーがロールのメンバーシップを制御できるようになります。

制約事項

ロールの使用に関しては、次のようないくつかの制約事項があります。

- ロールはデータベース・オブジェクトを所有できません。
- ロールには、セキュリティー管理者 (SECADM) 権限を付与できません。
- 以下のデータベース・オブジェクトの作成時には、グループに付与された許可およびロールは検討の対象にはなりません。
  - 静的 SQL を格納するパッケージ。
  - ビュー
  - マテリアライズ照会表 (MQT)
  - トリガー
  - SQL ルーチン

  オブジェクトを作成するユーザーに対してか、または PUBLIC に対して直接または間接的に (例えばロール階層を介して) 付与されたロールだけが、上記のオブジェクトの作成時に検討の対象になります。

付録 F. トラステッド・コンテキストおよびトラステッド接続

トラステッド・コンテキストとは、データベースと外部エンティティー (アプリケーション・サーバーなど) の間の接続における信頼関係を定義するデータベース・オブジェクトのことをいいます。

信頼関係は、以下の属性のセットに基づいています。

- システム許可 ID: データベース接続を確立するユーザーを表します
- IP アドレス (またはドメイン・ネーム): データベース接続を確立するホストを表します
- データ・ストリーム暗号化: データベース・サーバーとデータベース・クライアントの間のデータ通信のための暗号化設定がある場合にはそれを表します

ユーザーがデータベース接続を確立するときに、DB2 データベース・システムは、接続がデータベース内のトラステッド・コンテキスト・オブジェクトの定義と一致するかどうかを検査します。一致していた場合、データベース接続は信頼できると見なされます。

トラステッド接続を使用すると、このトラステッド接続の起動側では、トラステッド接続の有効範囲外では使用できない追加機能を取得することができます。追加機能は、トラステッド接続が明示的であるか暗黙的であるかによって異なります。

明示的トラステッド接続の起動側には以下の機能があります。

- その接続の現行ユーザー ID を、認証のあるなしに関係なく別のユーザー ID に切り替える
- トラステッド・コンテキストのロール継承フィーチャーにより追加の特権を取得する

暗黙的トラステッド接続は、明示的に要求されていないトラステッド接続で、明示的トラステッド接続要求ではなく通常の接続要求により確立されます。暗黙接続を取得するためにアプリケーション・コードを変更する必要はありません。また、暗黙的トラステッド接続を取得するかどうかは、接続戻りコードには影響はありません (明示的トラステッド接続を要求する場合は、接続戻りコードは要求が成功したかどうかを示します)。暗黙的トラステッド接続の起動側は、トラステッド・コンテキストのロール継承フィーチャーにより追加の特権を取得できるだけで、ユーザー ID を切り替えることはできません。

トラステッド・コンテキストを使用するとどのようにセキュリティーが向上するか

3 層アプリケーション・モデルは、クライアント・アプリケーションとデータベース・サーバーの間に中間層を置くことにより、標準的な 2 層クライアントおよびサーバー・モデルを拡張します。このモデルは、Web ベースのテクノロジーや Java™ 2 Enterprise Edition (J2EE) プラットフォームの登場により、近年特に大きな人気を得ています。3 層アプリケーション・モデルをサポートするソフトウェア・プロダクトの一例として、IBM WebSphere Application Server (WAS) があります。

© Copyright IBM Corp. 2007, 2008

3 層アプリケーション・モデルでは、クライアント・アプリケーションを実行するユーザーの認証、およびデータベース・サーバーとの相互作用の管理は、中間層が処理します。従来の方法では、データベース・サーバーとのすべての相互作用は、データベース・サーバーに対して中間層を識別するユーザー ID と資格情報の組み合わせを使用して、その中間層により確立されたデータベース接続を介して行われます。言い換えると、データベース・サーバーは、中間層のユーザー ID に関連付けられたデータベース特権を使用して、すべてのデータベース・アクセスで行う必要がある許可検査および監査を行います。これには、ユーザーの代わりに中間層により実行されるアクセスも含まれます。

3 層アプリケーション・モデルには多くの利点がありますが、データベース・サーバーとのすべての相互作用 (例えば、ユーザー要求) を中間層の許可 ID で行うようにすると、いくつかのセキュリティー上の問題が生じます。これらを要約すると以下のようになります。

- ユーザー ID の消失

  企業によっては、アクセス制御の目的で、データベースにアクセスしている実際のユーザーの ID を知りたい場合があります。

- ユーザーの説明責任の減少

  監査による説明責任は、データベース・セキュリティーにおける基本原則です。ユーザーの ID が不明であると、中間層の固有の目的のために中間層により実行されるトランザクションと、ユーザーのために中間層により実行されるトランザクションを区別することが難しくなります。

- 中間層の許可 ID に特権を付与しすぎる

  中間層の許可 ID には、すべてのユーザーからのすべての要求を実行するために必要なすべての特権が含まれていなければなりません。これには、特定の情報にアクセスする必要がないユーザーがアクセス権限を取得できてしまうというセキュリティー問題があります。

- 弱いセキュリティー

  前述の特権の問題に加えて、現行方式では、接続するために中間層により使用される許可 ID には、ユーザー要求によりアクセスされる可能性があるすべてのリソースへの特権を付与する必要があります。中間層の許可 ID の暗号漏えいが発生すると、それらすべてのリソースは公開されてしまいます。

- 同じ接続を使用するユーザー間で相互に影響を及ぼす

  直前のユーザーによる変更が現行ユーザーに影響を与える場合があります。

明らかに、実際のユーザーの ID およびデータベース特権が、そのユーザーに代わって中間層により実行されるデータベース要求で使用されるようなメカニズムが必要です。この目標を達成するための最も簡単な方法は、中間層がユーザーの ID とパスワードを使用して新規接続を確立した後、ユーザーの要求をその接続を介して送信するというものです。この方法は単純ですが、以下に挙げるようないくつかの欠点があります。

- 特定の中間層では不適当。多くの中間層サーバーには、接続を確立するために必要なユーザー認証資格情報がありません。

- パフォーマンス上のオーバーヘッド。新しい物理接続を作成し、データベース・サーバーでユーザーを再認証することに関連した、パフォーマンス上の明らかなオーバーヘッドがあります。
- 保守上のオーバーヘッド。一元的なセキュリティー・セットアップ、またはシングル・サインオンを使用していない状況では、2 つのユーザー定義 (1 つは中間層上、もう 1 つはサーバー上) を持つことによる保守上のオーバーヘッドがあります。この状況では、異なる場所にあるパスワードを変更することが必要です。

トラステッド・コンテキスト機能は、この問題を解決します。セキュリティー管理者は、データベースと中間層の間の信頼関係を定義するトラステッド・コンテキスト・オブジェクトをデータベースに作成できます。その後、中間層ではデータベースへの明示的トラステッド接続を確立できますが、この接続では、接続の現行ユーザー ID を、認証のあるなしに関係なく別のユーザー ID に切り替える機能が中間層に付与されます。トラステッド・コンテキストは、エンド・ユーザーの ID アサーション問題を解決するだけでなく、別の利点もあります。それは、データベース・ユーザーが特権を使用できるようになる時期を制御する機能です。ユーザーが特権を使用できる時期を制御できないと、全体的なセキュリティーの低下につながります。例えば、特権が、最初に意図した目的以外で使用される場合があります。セキュリティー管理者は、1 つ以上の特権を 1 つのロールに割り当て、そのロールをトラステッド・コンテキスト・オブジェクトに割り当てることができます。そのトラステッド・コンテキストの定義と一致するトラステッド・データベース接続 (明示的または暗黙的) のみがそのロールに関連付けられた特権を利用できます。

パフォーマンスの向上

トラステッド接続を使用すると以下の利点があるため、パフォーマンスを最大限に発揮します。

- 接続の現行ユーザー ID が切り替わる時に新規接続は確立されません。
- トラステッド・コンテキスト定義が切り替え先のユーザー ID の認証を必要としない場合には、データベース・サーバーで新規ユーザーを認証することに関連したオーバーヘッドは発生しません。

トラステッド・コンテキストの作成例

セキュリティー管理者が以下のトラステッド・コンテキスト・オブジェクトを作成すると想定します。

```
CREATE TRUSTED CONTEXT CTX1
  BASED UPON CONNECTION USING SYSTEM AUTHID USER2
  ATTRIBUTES (ADDRESS   '192.0.2.1')
  DEFAULT ROLE managerRole
   ENABLE
```

ユーザー *user1* が IP アドレス 192.0.2.1 からトラステッド接続を要求した場合、DB2 データベース・システムは、トラステッド接続を確立できなかったためユーザー *user1* が非トラステッド接続を取得したことを示す警告 (SQLSTATE 01679、SQLCODE +20360) を戻します。しかし、ユーザー *user2* が IP アドレス 192.0.2.1 からトラステッド接続を要求した場合には、接続属性はトラステッド・コンテキスト CTX1 により条件が満たされるため、要求は受け入れられます。ユーザー *user2* はトラステッド接続を確立したため、そのユーザーはトラステッド・コンテキストのロール managerRole に関連付けられたすべての特権および権限を取得で

きます。このトラステッド接続の有効範囲外では、ユーザー *user2* はこれらの特権および権限を使用できません。

付録 G. 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。

© IBM Corporation 1993, 2008

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。英語の DB2 バージョン 9.5 のマニュアル (PDF 形式) とその翻訳版は、 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 141. 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
管理 API リファレンス	SC88-4431-01	入手可能
管理ルーチンおよびビュー	SC88-4435-01	入手不可
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	SC88-4433-01	入手可能
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	SC88-4434-01	入手可能
コマンド・リファレンス	SC88-4432-01	入手可能
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	SC88-4421-01	入手可能
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	SC88-4423-01	入手可能
データ・サーバー、データベース、およびデータベース・オブジェクトのガイド	SC88-4259-01	入手可能
データベース・セキュリティー・ガイド	SC88-4418-01	入手可能
ADO.NET および OLE DB アプリケーションの開発	SC88-4425-01	入手可能
組み込み SQL アプリケーションの開発	SC88-4426-01	入手可能
Java アプリケーションの開発	SC88-4427-01	入手可能
Perl および PHP アプリケーションの開発	SC88-4428-01	入手不可
SQL および外部ルーチンの開発	SC88-4429-01	入手可能
データベース・アプリケーション開発の基礎	GC88-4430-01	入手可能

表 141. 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
DB2 インストールおよび管理概説 (Linux および Windows 版)	GC88-4439-01	入手可能
国際化対応ガイド	SC88-4420-01	入手可能
メッセージ・リファレンス 第 1 巻	GI88-4109-00	入手不可
メッセージ・リファレンス 第 2 巻	GI88-4110-00	入手不可
マイグレーション・ガイド	GC88-4438-01	入手可能
Net Search Extender 管理およびユーザーズ・ガイド	SC88-4630-01	入手可能
パーティションおよびクラスタリングのガイド	SC88-4419-01	入手可能
Query Patroller 管理およびユーザーズ・ガイド	SC88-4611-00	入手可能
IBM データ・サーバー・クライアント機能 概説およびインストール	GC88-4441-01	入手不可
DB2 サーバー機能 概説およびインストール	GC88-4440-01	入手可能
Spatial Extender and Geodetic Data Management Feature ユーザーズ・ガイドおよびリファレンス	SC88-4629-01	入手可能
SQL リファレンス 第 1 巻	SC88-4436-01	入手可能
SQL リファレンス 第 2 巻	SC88-4437-01	入手可能
システム・モニター ガイドおよびリファレンス	SC88-4422-01	入手可能
問題判別ガイド	GI88-4108-01	入手不可
データベース・パフォーマンスのチューニング	SC88-4417-01	入手可能
Visual Explain チュートリアル	SC88-4449-00	入手不可
新機能	SC88-4445-01	入手可能
ワークロード・マネージャー ガイドおよびリファレンス	SC88-4446-01	入手可能
pureXML ガイド	SC88-4447-01	入手可能
XQuery リファレンス	SC88-4448-01	入手不可

表 142. 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
DB2 Connect Personal Edition 概説およびインストール	GC88-4443-01	入手可能

表 142. DB2 Connect 固有の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能かどうか
DB2 Connect サーバー機能 概説およびインストール	GC88-4444-01	入手可能
DB2 Connect ユーザーズ・ガイド	SC88-4442-01	入手可能

表 143. 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
Information Integration: フェデレーテッド・システム 管理ガイド	SC88-4166-01	入手可能
Information Integration: レプリケーションおよびイベント・パブリッシングのための ASNCLP プログラム・リファレンス	SC88-4167-02	入手可能
Information Integration: フェデレーテッド・データ・ソース構成ガイド	SC88-4185-01	入手不可
Information Integration: SQL レプリケーション ガイドおよびリファレンス	SC88-4168-01	入手可能
Information Integration: レプリケーションとイベント・パブリッシング 概説	GC88-4187-01	入手可能

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/v9r5) 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 の「ご注文について」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、336 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 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 状態ヘルプを呼び出すには、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 9.5 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らない場合があります。

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。非管理者および非 root の DB2 インフォメーション・センター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。インストールする更新がある場合は、「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は、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する必要があります。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 Vista の場合、下記の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・ツールを起動するには、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/db2icdv95 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 9.5 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/V9.5 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 ブラウザーが起動し、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。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終了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終了しないでください。
   - 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終了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 スクリプトを終了しないでください。
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/db2icdv95 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 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「サポートおよびトラブルシューティング」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 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/udb/support.html) 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 H. 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。

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。

本書は、IBM 以外の Web サイトおよびリソースへのリンクまたは参照を含む場合があります。 IBM は、本書より参照もしくはアクセスできる、または本書からリンクされた IBM 以外の Web サイトもしくは第三者のリソースに対して一切の責任を負いません。 IBM 以外の Web サイトにリンクが張られていることにより IBM が当該 Web サイトを推奨するものではなく、またその内容、使用もしくはサイトの所有者について IBM が責任を負うことを意味するものではありません。 また、IBM は、お客様が IBM Web サイトから第三者の存在を知ることになった場合にも (もしくは、IBM Web サイトから第三者へのリンクを使用した場合にも)、お客様と第三者との間のいかなる取引に対しても一切責任を負いません。 従って、お客様は、IBM が上記の外部サイトまたはリソースの利用について責任を負うものではなく、また、外部サイトまたはリソースからアクセス可能なコンテンツ、サービス、

© IBM Corporation 1993, 2008

製品、またはその他の資料一切に対して IBM が責任を負うものではないことを承諾し、同意するものとします。 第三者により提供されるソフトウェアには、そのソフトウェアと共に提供される固有の使用条件が適用されます。

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。

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

以下は、International Business Machines Corporation の米国およびその他の国における商標です。

pureXML	Redbooks
z/OS	developerWorks
ibm.com	DB2
IBM	AIX

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。
- Windows 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]

© Copyright IBM Corp. 2007, 2008

［サ行］

［タ行］

［ナ行］

［ハ行］

[マ行]

[ラ行]

[ワ行]

A

Printed in Japan

SC88-4446-01

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:

DB2 Version 9.5 for Linux, UNIX, and Windows

ワークロード・マネージャー ガイドおよびリファレンス