# DB2 モニタリング技法
## -あなたの DB をうまくモニタリングする方法-

たかやま　ひろあき
高山　　博章

株式会社エクサ　　基盤ソリューション本部
ＩＴプロフェッショナル

原稿量
本文　　7,300 字
要約　　1,000 字
図表　　11 枚

# ＜要　約＞

近年複数のデータベースの統合やそもそものデータ容量の増加に伴い、データベースの規模も増加の一途をたどっている。それに伴い今までは別々のシステムで稼働していたさまざまなデータベース処理（アプリケーション）が一つのデータベース・システムで稼働するようになった。このため、パフォーマンスのトラブルが発生した際の原因もその種類や発生箇所がさまざまであることが多い。このような問題の原因を特定し、本来あるべき性能をデータベースが発揮できるようにするというのがパフォーマンス問題における問題判別の目的である。本論文では DB2 においてパフォーマンス問題が発生した場合に備えて、数あるモニタリング技法から日々適切なモニタリングを行う方法、適切な方法で問題解決を行う方法を述べる。

パフォーマンス問題が発生した際には、現在の状態が平常時と異なっているかどうかを判断するためには、日々モニターを行い、平常時の状態をモニターしておく必要がある。日々のモニターを行っていないと問題発生時に、現在の状態がどう問題があるのか判断できない。また、問題発生時には適切なモニター機能を選択しないと問題解決にたどりつけない、時間がかかることになる。上記より、日々のモニターの蓄積、適切なモニター機能を選択することは非常に重要になる。

DB2 の提供するモニター機能を使用し、日々のモニター・データを蓄積する方法としては、ファイル形式とデータベース内の表形式の保管を選択することができる。ファイル形式と表形式の保管を比較すると、一回限りのモニター・データの取得には、結果の確認しやすさでファイル形式が向いている。逆に複数回のモニター結果の確認の場合には、集計や汎用的な分析などを SQL で簡単に行うことができるため表形式の保管が有効である。ファイル形式の場合分析ごとに処理を作りこむ必要が出る。また、運用面においても表形式の方がデータベースの機能に任せることができるため有利な点が多い。例えば保管期間を過ぎたデータの削除は SQL で行うことができるし、バックアップに関してもモニター・データはデータベース内の表に保管しているため、データベースのバックアップに任せることができる。

実際に表形式においてモニター・データを蓄積するための表の定義、SQL を使用したモニター・データの取得方法といった実装方法を説明する。また、接続アプリケーション情報、バッファープール情報を使用し、いくつかの SQL を使用したモニター・データの分析例をあげ、汎用分析において表形式での保管がファイル形式と比べて優れているかを説明する。

# 目　次

# 1.はじめに

ひとくちにDB2におけるパフォーマンスのトラブルといっても、その種類や発生箇所はさまざまである。またその問題の原因も、単純なものから複雑なものまで多くのケースがある。多様な機能が複雑に連携しあって稼働しているデータベース・サーバーを少しずつひもとき、問題発生の原因箇所を突き止めて解消し、本来あるべき性能をDB2が発揮できるようにしたい、というのがパフォーマンス問題における問題判別の目的である。

本論文では、日々モニタリングをすることの重要性、数あるモニタリング技法からパフォーマンス問題が発生時に備えて日々適切なモニタリングをする方法、及び問題発生時に適切なモニタリング技法を使用して、問題解決を行う方法を,実例を交えて解説を行う。

# 2.モニタリングの重要性

データベース・システムにおいて、日々データベースの稼働状況をモニターし、その結果を保管しておくことは非常に重要である。

例えばデータベース・システムにおいて、パフォーマンスの問題が発生した場合を考える。パフォーマンスの問題が発生した場合、既に問題となる処理(SQL)が特定されている場合や漠然とシステム全体が遅くなっている場合など状況はさまざまである。ただし、どのような状況下においてもSQLのアクセス・パスを確認あるいは、データベースのモニター・データを取得して、今現在の状況を把握する必要が出てくる。そして、取得したデータからパフォーマンスの問題となる箇所を特定することになるが、日々の平常時にモニター・データを取得していない場合、今現在のモニター・データ値からだけで問題を特定する必要が出てくるが、このシステムでの指標となるデータがないため、問題を特定するためにはデータベースに関する高いスキルも持ち合わせたエンジニアが必要になる。現場にそのようなエンジニアがいない場合、問題の特定は困難をきわめることが多々ある。最悪の場合、問題解決にたどりつくことができないという事態に陥ることになる。

逆に常日ごろ、モニター・データを取得し保管している場合、平常時と問題発生時の比較を行うことが可能になるため、平常時とどこが異なるのかということを容易に特定することが可能になる。これにより平常時との差異の箇所がデータベース的にどのようなことで発生するかを調査すればよいことになるので、早期に問題解決にたどりつけるようになる。

また、問題解決に向けて日々のモニター・データの蓄積以外にももう一つ大切なことがある。それは有効なモニター･データを蓄積することである｡DB2には複数のモニター機能が提供されており、それらは使用する場面、用途が異なっている。使用する側はこれらのモニター機能をじゅうぶんに理解し、シチュエーションごとに適切なモニター機能を使用する必要がある。各モニター機能を十分じゅうぶんに理解しないまま使用すると、せっかく日々蓄積したデータが使い物にならなく、問題解決の糸口を見つけることができなくなってしまう。

上記のことから分かるように、データベース・システムにおいて各種モニター機能を正確に理解し使用すること、そして日々モニタリングを行い、データを蓄積することは非常に重要である。これらの仕組みをシステム構築時にきちんと作ることによって、本番稼働後も問題発生の有無によらず、

モニター・データを有効活用することが可能になる。

以降、DB2 が提供するモニター機能の説明、モニター・データの蓄積方法、モニター・データの有効な活用方法について説明する。

## 3. 適切なモニタリング機能の選択

DB2 はデータベースをモニタリングするために、複数のモニター機能を提供している。この章では各モニター機能の説明と多くの場面で有効なスナップショット・モニターとメトリック・モニターについての機能を比較し、具体的にどのような場面で使用すべきかを説明する。

### 3.1 DB2 が提供するモニター機能

ここでは、DB2 が提供しているモニター機能について簡単に説明をする。詳細な機能、使用方法については、IBM DB2 Version 10.5 Information Center を参照いただきたい。各モニター機能についての特徴を表 1 にまとめた。

表 1. モニター機能の特徴

| | スナップショット・モニター | | メトリック・モニター | | イベント・モニター | |
|---|---|---|---|---|---|---|
| | snapshot コマンド | snapshot 表関数 | メトリック表関数 | MONREPORT | ファイル保管 | 表保管 |
| 実行方法 | コマンド | SQL | SQL | SQL（ストアド・プロシージャ） | SQL | SQL |
| データ鮮度 | 実行時の情報 | 実行時の情報 | 実行時の情報 | 実行時の情報（間隔指定可） | 開始後からのすべての情報 | 開始後からのすべての情報 |
| 出力結果 | ファイル | 表 | 表 | ファイル | ファイル | 表 |
| 分析・集計 | 作りこみが必要 | SQL | SQL | 作りこみが必要 | 作りこみが必要 | SQL |
| 負荷 | 中 | 中 | 低 | 低 | 高 | 高 |

表 1 にまとめた以外にも DB2 はモニター機能を提供しているが使用頻度が少ないため、本論文では割愛する。表 1 にあげた各モニター機能はスナップショット・モニター/メトリック・モニターとイベント・モニターに大きく分類される。前者は DB2 コマンドまたは SQL 実行時の情報が取得され、後者は開始後すべての情報が取得される。前者はデータベースへの負荷もそれほど高くなく、定期的に実行しデータベースの稼働状況を監視/分析するのに適している。また、パフォーマンスの問題が発生した際にも、即座に状況を確認することが可能である。後者はデータベースへの負荷が高いため、すべての情報を欠落なく取得したい場合、例えば監査目的で実行されたすべての SQL を取得したいといった用途にのみ取得すべきであり、一般的に定常情的にモニターすべきものではない。このため、

本論文でもイベント・モニターの活用方法については論述しない。

### 3.2 スナップショット・モニターとメトリック・モニターの比較

まず、スナップショット・モニターとメトリック・モニターの違いについて説明する。スナップショット・モニターとメトリック・モニターは、取得可能な項目が異なるが両者とも実行時の情報を取得する。メトリック・モニターは SQL でのみ実行可能であるが、スナップショット・モニターは SQL 以外にも DB2 コマンドでの実行も可能である。また、データベースに対する負荷という面では、メトリック・モニターの方が負荷が軽い。

次にスナップショット・モニターの出力形式について説明する。スナップショット・モニターでは DB2 コマンドで実行した場合と SQL で実行した場合で出力形式が異なる。DB2 コマンドでは出力はファイル形式になり、SQL では表形式になる。表 2 に出力形式の違いについてまとめた。

表 2. スナップショット・モニターの出力形式比較

| | ファイル形式 | 表形式 |
|---|---|---|
| 1 回の実行結果 | ◎<br>見やすい | ○<br>見づらい場合あり |
| 蓄積データの分析 | △<br>作りこみが必要 | ◎<br>容易 |
| 運用面 | △<br>作りこみが必要 | ◎<br>容易 |

スナップショット・モニターは、出力結果を DB2 コマンドによるファイルと SQL による表形式が選択可能である。モニター結果を確認及び分析するにあたって、ファイルと表出力の違いを比較する。一回の実行に限るのであれば、DB2 コマンドによるファイル出力の方が各モニター項目も表示され、SE がモニター結果の確認が容易である。SQL による表形式の場合、モニター値が 1 行に表示されることになるため全モニター値を確認しようとすると、見づらくなる。ただし、確認したいモニター項目が決まっているのであれば、SELECT で確認したい項目を選択することが可能なので、確認は容易である。一回の実行ではなく、複数回のモニター結果を確認及び分析する場合、ファイル形式だと複数ファイルから分析するための処理（プログラムなど）を別途作りこむ必要が出てくる。また、さまざまな切り口で分析を行おうとするとそのレポート数分の処理の作成が必要になり手間がかかる。次に、分析手法が確立されていればよいが、そうでない汎用的な分析を行う必要が出ると、せっかく有益なモニター・データを蓄積したにも関わらず、分析自体が不可能になってしまう。表形式の場合、複数回のモニター結果はデータベース内の表にそのまま蓄積することが可能である。このため、ファイル形式のように複数回のモニター結果を確認及び、分析する場合、結果は表に蓄積されているので複雑な集計や必要な項目だけの選択などが SQL で可能であり確認/分析が容易に行うことができる。また、蓄積したモニター結果の運用面（メンテナンス）で比較する。ファイル形式、表形式とも蓄積するモニター結果バックアップを定期的に行うことと古くなった結果を削除することが必要になる。ファイル形式の場合、ファイルシステムにファイルとして保管されることになるが、それらを定期的にバックアップ、削除を行う処理を別途作成する必要が出てくる。逆に表形式での保管の場合、モニター結果はデータベース内の保管されているため、バックアップに関してはデータベースのバックアップを

行うことによって自動的にバックアップされるので、別途処理を作成する必要がない。削除に関しては SQL により DELETE を行うことによって実装可能である。このことから運用面に関しても表形式の方がメリットが大きい。

以上から、ファイル形式と表形式の保管を比較すると一回限りの実行結果の場合、DB2 コマンドによるファイル形式の方が見やすく扱いやすいので、問題発生時などでその瞬間を確認したい場合にはファイル形式が容易であるので、こちらの手法を選択すべきである。蓄積されたモニター結果を深く分析及び運用面を考慮するのであれば、表形式で保管をしていた方が分析しやすく、問題の原因を特定することが容易である。また運用もデータベースの機能に任すことができるので SQL でのモニタリングを選択すべきである。

以降、表形式でモニター結果を保管し蓄積する仕組み、及び蓄積したモニター・データを SQL によって分析する方法について説明する。

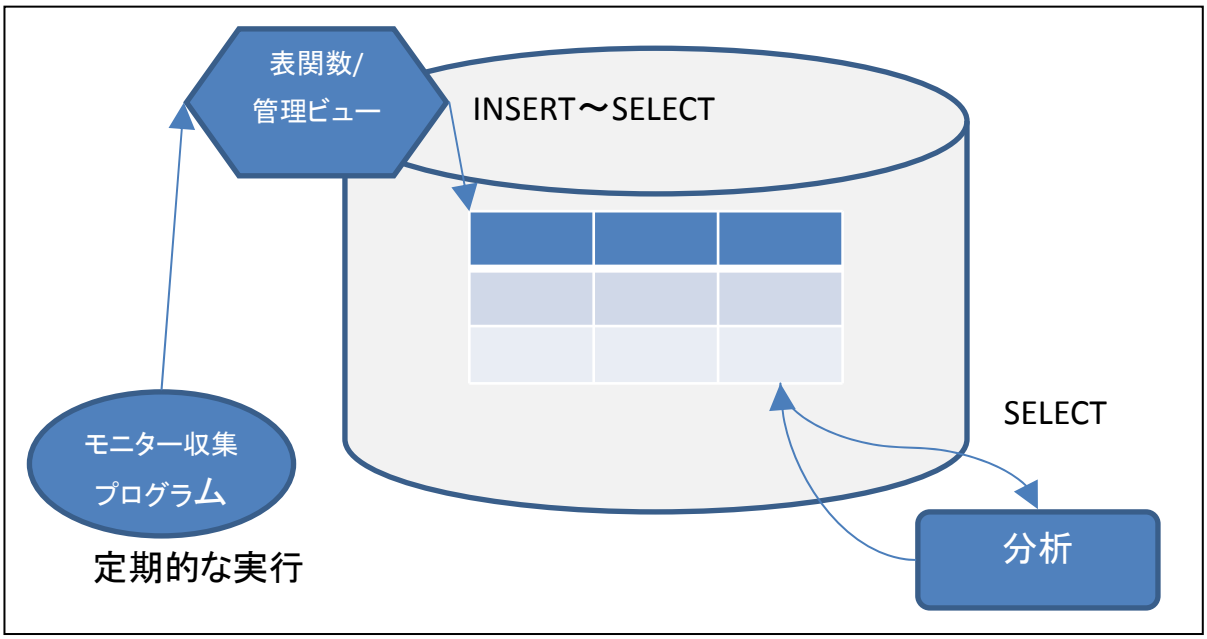
## 4. モニタリング・データの活用方法

ここでは前述したモニター機能を使用し、日々のモニタリング・データをデータベース内の表に蓄積する仕組みとそのモニタリング・データを分析するための SQL を実例を交えて説明する。

### 4.1 モニタリング・データを保管する仕組み

ここでは、前述した SQL ベースでデータベース内にモニター結果を保管し、蓄積したモニター・データを分析する仕組みについて説明する。図 1 はデータベース内にモニター・データを保管する仕組みの概略である。

図 1. データベース内にモニター・データを保管する仕組み




まず、図 1 のようにデータベース内にモニター用の表を定義する。データベース内格納する表のレイアウト（データ・タイプ）はモニターを行うために使用する表関数または管理ビューの出力を元

に定義を行う。表関数、管理ビューの出力の情報は IBM DB2 Version 10.5 Information Center を参照いただきたい。表 3 は、後述で事例説明する際に使用する表関数、管理ビューについてまとめたものである。

表 3. 表関数/管理ビュー

| 表関数名/管理ビュー名 | モニター内容 |
|---|---|
| スナップショット表関数/管理ビュー | |
| APPLICATIONS | 接続アプリケーションの情報 |
| SNAPAPPL | 接続アプリケーションの方法 |
| メトリック表関数 | |
| MON_GET_BUFFERPOOL | バッファープールの情報 |

表 3 にあげた表関数、管理ビューを使用して実際にモニター・データを表に保管する方法は、以下の図 2 に記載した SQL 文を定期的に実行することによって行う。

```
INSERT INTO モニター表 SELECT ～ FROM 表関数/管理ビュー
```

図 2. モニター・データを保管する SQL

実際に蓄積したモニター・データは SELECT を使用して分析を行うことになる。

以降、表 3 にあげた表関数及び管理ビューを使用したモニター方法と表に蓄積したモニター・データを使用して問題解決を行うための分析方法を、実例を交えて説明する。

### 4.2 蓄積したモニター・データの分析例

まず、データベースに接続しているアプリケーションの情報を分析する方法を説明する。アプリケーションのモニター情報を格納する表を表 4 のように定義を行う。

表 4. アプリケーション情報を格納する表定義(appl_info)

| 列名 | 内容 |
|---|---|
| SNAPSHO_TIMESTAMP | モニター時刻 |
| APPL_NAME | アプリケーション名 |
| AUTHID | 接続ユーザー |
| TOTAL_CPU_TIME | 使用 CPU 時間 |
| APPL_STATUS | アプリケーション状況 |

図 2 のような SQL を使用して定期的にモニター結果を表に保管する。実際の分析例として、二つあげる。一つ目の分析例は、時間ごとの接続しているアプリケーションの情報一覧を取得するものである。図 3 は表 4 の表に対して、分析 SQL を実行した結果を表したものである。

```
select snapshot_timespamp,appl_name,authid,total_cpu_time,appl_status
from appl_info order by 1

SNAPSHOT_TIMESTAMP         APPL_NAME AUTHID  TOTAL_CPU_TIME APPL_STATUS
-------------------------- --------- ------- -------------- -------------
2013-12-10-18.45.01.958745      db2bp  USER00       0.011444 UOWWAIT
2013-12-10-18.45.01.958745      db2bp  USER00       0.000295 UOWEXEC
2013-12-10-18.45.01.958745       java  USER02       0.016798 CONNECTED
2013-12-10-18.45.01.958745       java  USER02       0.010315 UOWWAIT
2013-12-10-18.45.01.958745     aa.exe  USER01       0.012255 UOWWAIT
2013-12-10-18.45.01.958745     aa.exe  USER01       0.014551 UOWWAIT
2013-12-10-18.45.01.958745      db2bp  USER00       0.479227 UOWWAIT
2013-12-10-18.45.01.958745      db2bp  USER02       0.026994 UOWWAIT
2013-12-10-18.55.01.987458      db2bp  USER00       0.003419 UOWEXEC
2013-12-10-18.55.01.987458       java  USER02       0.016798 UOWWAIT

10 レコードが選択されました。
```

図 3.アプリケーション情報一覧を取得する例

簡単な SQL 文によって、接続アプリケーション情報の一覧が取得できることが分かる。これにより、問題発生時に接続していたアプリケーションがどの程度 CPU を使用していてたかやどのような状態（実行中、ロック待ちなど）にあったかを判断することができる。

二つ目の分析例として、データベースに接続しているユーザー数の時間遷移を分析するものをあげる。図 4 は一つ目の例と同じく分析 SQL を実行した結果を表したものである。

```
select time(snapshot_timestamp) as time,
sum(case when authid='USER00' then 1 else null end) as USER00,
sum(case when authid='USER01' then 1 else null end) as USER01,
sum(case when authid not in ('USER00','USER01') then 1 else null end) as OTHERS
from appl_info
where date(snapshot_timestamp) = current date
group by time(snapshot_timestamp) order by 1

TIME     USER00      USER01      OTHERS
-------- ----------- ----------- -----------
00:00:03           2           -           -
00:05:02           1           -           -
・・・    ・・・       ・・・       ・・・
08:00:02           1           5           -
08:05:02         439          10           2
・・・    ・・・       ・・・       ・・・
23:50:02          25           -           -
23:55:02          49           -           -

  201 レコードが選択されました。
```

図 4.ユーザー数の時間遷移を取得する例

少し分析用の SQL は複雑にはなるが、データベースへの接続ユーザー数の推移を確認することができる。この分析を表形式ではなく、ファイル形式のモニターで行うとするとかなりの作りこみの処理を作成する必要が出てくる。この例は表形式でモニター結果を保管する場合の有利な点といえる。

次にデータベースのモニターで非常に重要な項目となるバッファープールのヒット率を分析する方法を説明する。バッファープールのモニター情報を格納する表を表 5 のように定義を行う。

表 5.バッファープール情報を格納する表定義(bp_info)

| 列名 | 内容 |
|---|---|
| SNAPSHO_TIMESTAMP | モニター時刻 |
| BP_NAME | バッファープール名 |
| POOL_L_READS | 論理読み取り数 |
| POOL_P_READS | 物理読み取り数 |
| HITRATIO | バッファープールヒット率 |

前述したアプリケーションの分析例と同様にバッファープール情報の分析も、二つ例をあげる。

一つ目の分析例は、時間ごとのバッファープールのヒット率情報一覧を取得するものである。図５は表５の表に対して、分析 SQL を実行した結果を表したものである。

```
select mon_timestamp,bp_name,pool_l_reads,pool_p_reads,hitratio
from bp_info order by 1

MON_TIMESTAMP              BP_NAME          POOL_L_READS         POOL_P_READS         HITRATIO
-------------------------- ---------------- -------------------- -------------------- --------
2013-12-30-11.27.37.308000 IBMDEFAULTBP                  19850                  704    96.45
2013-12-30-11.28.20.442000 IBMDEFAULTBP                  19909                  713    96.41
2013-12-30-11.37.37.316000 IBMDEFAULTBP                  25002                  730    97.08
2013-12-30-11.52.17.847000 IBMDEFAULTBP                  39430                 1222    96.90
2013-12-30-11.53.28.172000 IBMDEFAULTBP                  39438                 1224    96.89
2013-12-30-11.54.01.790000 IBMDEFAULTBP                  50009                 1329    97.34

  6 レコードが選択されました。
```

図 5.バッファープールのヒット率一覧を取得する例

この分析例では簡単なSQLによって時間ごとのバッファープールのヒット率を確認することができる。問題発生時にバッファープールのヒット率に問題がなかったか、平常時と比べ違いがなかったなど即座に確認することが可能である。ただし、このモニター情報には問題点も存在する。バッファープールのヒット率は、データベース内の論理読み取り数と物理読み取り数から計算されるものであるが、各読み取り数はデータベースが開始されてからの積算値になっている。このため、データベースが開始されてから時間がたつと、時間当たりごとの処理に大きなブレが生じないとヒット率に変化がほとんど見られなくなってしまう。この問題を解消するために、モニター間隔ごとのバッファープールのヒット率を取得する SQL を図６に表す。

```
with hist_bp as
(select mon_timestamp,bp_name,pool_l_reads,pool_p_reads,hitratio,
dense_rank() over (order by mon_timestamp desc) as rank
from
bp_mon order by mon_timestamp desc),
diff_bp (mon_timestamp,bp_name,pool_l_reads,pool_p_reads,hitratio)
as
(select t1.mon_timestamp,t1.bp_name,
(t1.pool_l_reads - t2.pool_l_reads) as pool_l_reads,
(t1.pool_p_reads - t2.pool_p_reads) as pool_p_reads,
case when t1.pool_l_reads - t2.pool_l_reads > 0
 then  dec((1  -  (float(t1.pool_p_reads  -  t2.pool_p_reads)  /  float(t1.pool_l_reads  -
t2.pool_l_reads))) * 100,5,2)
 else null
end as hitratio
from
hist_bp t1,hist_bp t2
where t1.rank + 1 = t2.rank and t1.bp_name = t2.bp_name)
select mon_timestamp,bp_name,pool_l_reads,pool_p_reads,hitratio
from diff_bp
order by mon_timestamp

MON_TIMESTAMP              BP_NAME      POOL_L_READS          POOL_P_READS          HITRATIO
-------------------------- ------------ -------------------- -------------------- --------
2013-12-30-11.28.20.442000 IBMDEFAULTBP                   59                    9    84.74
2013-12-30-11.37.37.316000 IBMDEFAULTBP                 5093                   17    99.66
2013-12-30-11.52.17.847000 IBMDEFAULTBP                14428                  492    96.58
2013-12-30-11.53.28.172000 IBMDEFAULTBP                    8                    2    75.0
2013-12-30-11.54.01.790000 IBMDEFAULTBP                10571                  105    99.00

  25 レコードが選択されました。
```

図 6.時間ごとのバッファープールのヒット率一覧を取得する例

分析用の SQL は複雑になってしまうが、モニター間隔ごとのバッファープールのヒット率を確認することができる。これにより、一つ目の分析例では見逃してしまうような細かい時間内でのヒット

率の低下を確認することが可能になり、問題発生の要因を突き止めることができる。やはりアプリケーション情報の取得と同様に、この種の分析はファイル形式よりも表形式の保管の方が有利である。

本論文では、アプリケーション、バッファープールの観点での分析例を説明したが、ファイル形式ではなく、表形式でモニター・データを蓄積することによって、これ以外にもさまざまな分析が可能になると考えられる。

## 5.おわりに

本論文では、パフォーマンス問題などが発生した際に備え、日ごろからモニター・データを蓄積しておくことの重要性を説明した。また、モニター・データを保管する方法として、DB2 ではファイル形式と表形式の保管方法があるが、分析の観点や運用の観点で表形式の保管に有利な点が多いた表形式での保管を強く推奨する。本論文がデータベース・システムでのモニタリングにおいて少しでも役立てられることを願う。

参考文献

1. IBM DB2 Version 10.5 Information Center
URL: http://pic.dhe.ibm.com/infocenter/db2luw/v10r5/index.jsp

---