DB2 ユニバーサル・データベース

サポートハンドブック(AIX 版)

第 2 版[V8.1 対応]

日本アイ・ビー・エム株式会社
ソフトウェア事業
インフォメーション・マネジメント テクニカル・セールス

変更履歴:

2007/04/09 :

5-1-2 db2_kill の説明に、WSE（ワークグループ・サーバー・エディション）用の db2nkill の記述を追加しました。

目次

本書について

目的

「DB2 ユニバーサル・データベース for AIX サポートハンドブック(v8.1 対応)」は、DB2 の保守管理を行うお客様、ビジネス・パートナー様、技術サービス員が障害時に以下のことを容易に行うためのハンドブックです。

― 正確かつ短時間での障害の復旧

― 障害の原因特定を早期にはかるために必要な情報の収集

対象製品 :

AIX 版 DB2 UDB Enterprise Server Edition (ESE) V8. 1+ Fixpak4

- OS は AIX に特化した内容も記述しておりますが、基本的な考え方はその他のプラットホームにも適用可能です。
- DB2 Data Partitioning Feature に関する記述は含まれておりません。

対象読者 :

本書は、お客様、データベース管理者および DB2 クライアントおよびサーバーを担当する技術サービス員を対象としています。本書を使用するにあたって、以下の知識を保持されていることが望ましいです。

- DB2 グローバルマスターの DB2 エキスパート相当の知識
- ハードウェアおよびオペレーション・ソフトウェア全般に関する基礎知識
- ネットワーク全般に関する基礎知識

商標について

強調表示

本書では、次の強調表示規則を使用します。

Italics(斜体) 実際の名前または値をユーザーが指定する必要があるパラメーターを示します。

『』(鍵括弧) 参考文献、マニュアルを示します。

おことわり

本書で記載した内容は必ずしもユーザー様の実環境に適した内容ではございません。あくまでも、一つの目安として捉えていただき、実環境を把握したユーザー様の十分なご判断の元にご使用いただくようお願い申し上げます。

1章　回復管理、問題判別の基礎

1 データベースの回復管理

1-1 回復のために必要なもの

障害が発生したデータベースを回復するには、ログとデータベースのバックアップ・イメージが存在していることが基本となります。日頃よりログとデータベースのバックアップ・イメージの運用管理を確立しておくことが大切です。ここでは、ログとバックアップに関して特徴と使い方について説明します。

1-1-1 ログ

ログにはデータベース・オブジェクトとデータに対して行われた変更がすべて記録されています。

1-1-1-1 ログの状態

- アクティブ・ログ

  現在使用中のログ・ファイルで、クラッシュ・リカバリー(1-4-1 クラッシュ・リカバリー(破損回復)参 照) に使用します。アクティブ・ログにはコミット済みではあるがバッファープールからディスクのデータ領域に書き込まれていないトランザクションが記録されている場合があります。したがってアクティブ・ログは絶対に削除をしてはいけないログです。保管場所はデータベース構成パラメーターの LOGPATH の値で設定します。アクティブ・ログは万一の障害に備えてミラーリングすることも可能です(1-1-1-4 アクティブ・ログのミラーリング参照)。

- アーカイブログ

  アクティブ・ログがクローズされ、現在使用されていないログ・ファイルで、ロールフォワード・リカバリー(1-4-3 ロールフォワード・リカバリー参照)に使用します。将来のロールフォワード・リカバリーに備えて保存ログ方式(1-1-1-2 ログの方式参照)、USEREXIT プログラム(1-1-1-3 USEREXIT 参照)を用いるなどの方式で保存することをお奨めします。

1-1-1-2 ログの方式

ログの方式には循環ログ方式と保存ログ方式の2つがあります。

- 循環ログ方式

  ログ・ファイル内のトランザクションが全てコミットあるいはロールバックされるとログ・ファイルが再利用される方式です。データベース作成時のデフォルトのログ方式です。 循環ログ方式はロールフォワード回復を行うことはできませんが、破損回復、バージョン回復は可能です。

- 保存ログ方式(アーカイブ・ロギング)

  非活動になったログ・ファイルを将来の回復に備えて保存(アーカイブ)する方式です。(ロールフォワード回復を行うためにはこちらの方式を選択する必要があります。)

使用中のログ方式を確認する方法は以下のとおりです。

データベース構成パラメーターの LOGRETAIN 及び USEREXIT の値を確認します。

```
$ db2 get db cfg for <database-alias>  user <user-name> using <password>
```

<循環ログ方式の場合>

回復用ログの保持使用可能	(LOGRETAIN) = OFF
USER ロギング用ユーザー出口使用可能	(USEREXIT) = OFF

<保存ログ方式の場合>

回復用ログの保持使用可能	(LOGRETAIN) = RECOVERY
USER ロギング用ユーザー出口使用可能	(USEREXIT) = OFF

もしくは

回復用ログの保持使用可能	(LOGRETAIN) = OFF
USER ロギング用ユーザー出口使用可能	(USEREXIT) = ON

もしくは

回復用ログの保持使用可能	(LOGRETAIN) = RECOVERY
USER ロギング用ユーザー出口使用可能	(USEREXIT) = ON

1-1-1-3 USEREXIT

保存ログ方式で USEREXIT プログラムを用いることにより、アーカイブ・ログ・ファイルを任意の宛先に保存したり、また、回復時に必要なアーカイブ・ログ・ファイルを保存先より取り出したりすることができます。

USEREXIT プログラムはインスタンスあたり1つだけ作成出来ます。インスタンス内の複数のデータベースで USEREXIT プログラムを使用する時は、1つの USEREXIT プログラムで、対象となる全てのデータベースに対応できるようにコーディングを行って下さい。USEREXIT プログラムの実行形式ファイル名は変更できません。DB2 に付随しているサンプルの USEREXIT プログラムは C でご提供していますが、他の言語でも構いません。

USEREXIT を使うかどうかは、データベース構成パラメーターの USEREXIT で設定します。ON に設定すると、自動的に保存ログ方式となります。

1-1-1-4 アクティブ・ログのミラーリング

アクティブ・ログをミラーリングすると、以下の事態からデータベースを保護するのに役立ちます。

- アクティブ・ログの不慮の削除
- ハードウェア障害によるデータ破壊

アクティブ・ログが (ディスクが壊れた結果) 損傷するかもしれないことが心配な場合は、データベース構成パラメーター MIRRORLOGPATH を使用し、データベースの 2 次パスを指定してアクティブ・ログのコピーを管理することにより、ログの保管先のボリュームをミラーリングすることを考慮してください。MIRRORLOGPATH 構成パラメーターを使用すると、データベースは、ログ・ファイルの 2 つ目のコピー (同一の内容) を別のパスに書き込めます。物理的に別個のディスク (別のディスク・コントローラー上でもあることが望ましい) に 2 次ログ・パスを作成することをお勧めします。こうすれば、ディスク・コントローラーが単一の障害点になることはありません。

データベース構成パラメーターの MIRROLOGPATH の値を確認する方法は以下のとおりです。

```
$ db2 get db cfg for sample |find "MIRRORLOGPATH"
```

ログ・パスのミラー　　(MIRRORLOGPATH) =/mirrorlog /NODE0000/

1-1-2 バックアップ

DB2 の BACKUP コマンド、またはストレージ製品の高速コピー機能により、データベースのバックアップ・イメージを取得します。

1-1-2-1 バックアップの種類

バックアップは、モードや対象により、いくつかの観点に分かれます(オフライン／オンライン　データベース／表スペース フル／増分)。運用や回復要件により、どの組み合わせでバックアップを取得するか検討します。例えば、『週1回、オフラインでデータベース全体をフルでバックアップする』、などです。具体的な使用方法は1-3 節　バックアップ関連コマンド使用例を参照ください。

[a] オフライン・バックアップ／オンライン・バックアップ

オフライン・バックアップとはデータベースへの接続を全て切断してバックアップを取得するモードです。一方、オンライン・バックアップとは、バックアップ中に他のアプリケーションがデータベースに接続し作業を継続したままバックアップを取得できるモードです。オンライン・バックアップの前提のログ方式は保存ログ方式です。

[b] データベースのバックアップ／表スペースのバックアップ

データベースのバックアップとは、データベース全体をバックアップ対象としたものです。一方、表スペースのバックアップとは、指定した表スペースのみバックアップ対象としたものです。表スペース・バックアップの前提のログ方式は保存ログ方式です。

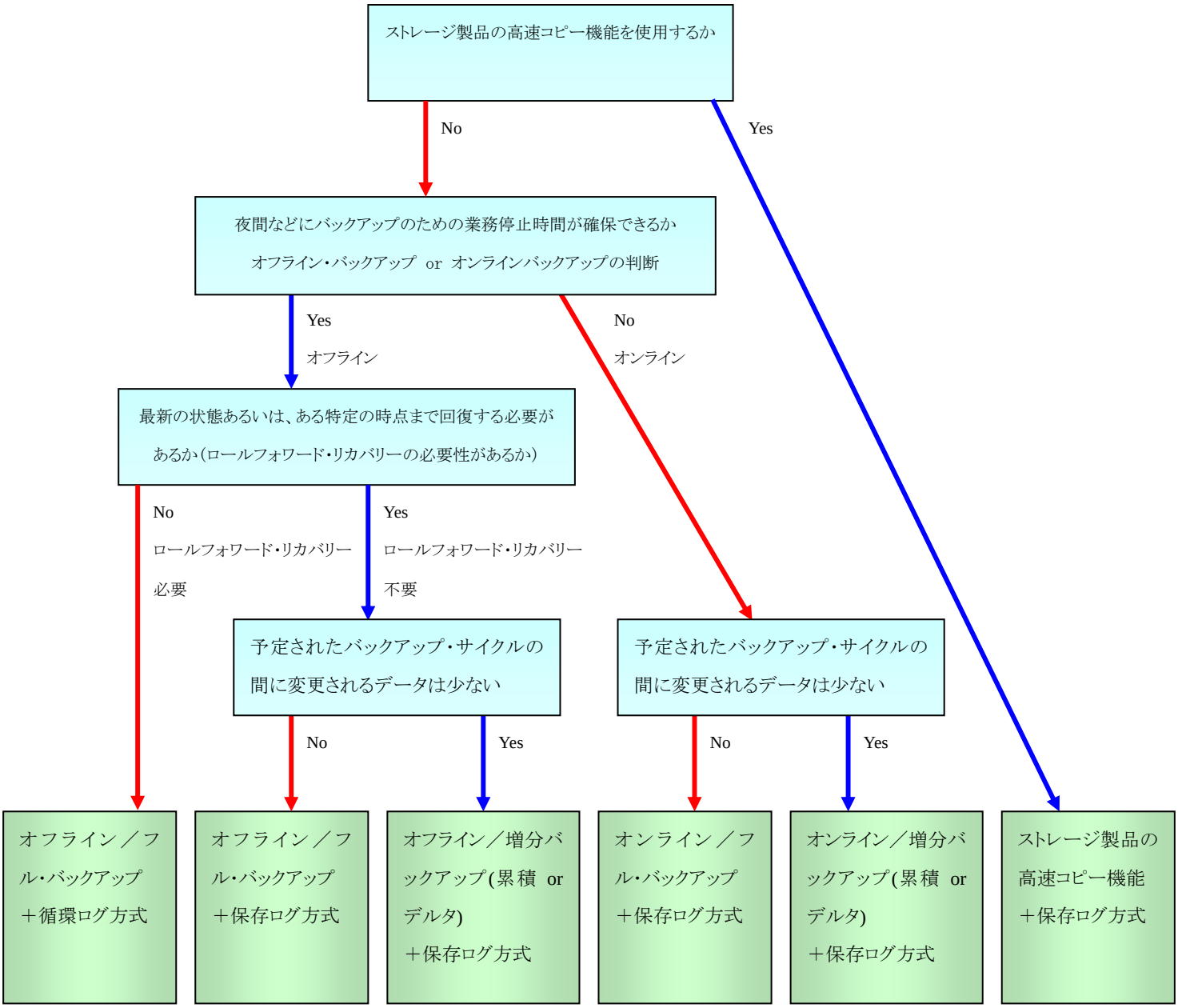
[c] フル・バックアップ／増分バックアップ

フル・バックアップは、全ページをバックアップ対象とします。一方、増分バックアップは、前回のバックアップ以降に更新されたページのみバックアップ対象とします。増分バックアップには、さらに累積タイプとデルタ・タイプ(非累積タイプ)の2タイプが選択できます。累積タイプとは正常終了した最新のフル・バックアップ以降の変更分を全て取得するもので、デルタ・タイプとは、該当表スペースの最新のフル・バックアップあるいは増分バックアップ（累積・デルタのいずれでも可）以降の変更分のみ取得するものです。

[d] ストレージ製品の高速コピー機能によるバックアップ

ストレージ製品の高速コピー機能(IBM ESS の PPRC など)により、データベースのバックアップを取得することができます。実動マシンからバックアップ操作の負荷を軽減し、高速にバックアップおよびリストアを実行することができます。

1-1-2-2 バックアップ種類選択のための目安



1-2 回復の概要

1-2-1 回復のタイプ

ログ、バックアップ、あるいはその両方を用いて行う回復処理には以下の 3 タイプがあります。

1-2-1-1 クラッシュ・リカバリー(破損回復)

データベースに対する作業単位が予期せず(電源障害・ソフトウェアの異常停止等)中断させられた場合にデータベースが不整合状態となります。再びトランザクションが処理可能な整合状態に戻すために、ACTIVE LOG を使用してデータベースを復帰させる回復です。

1-2-1-2 バージョン・リカバリー（復元回復）

DB2 の BACKUP コマンドで取得したバックアップ・イメージを使用して DB2 の RESTORE コマンドで復元してデータベースを回復させるものです。復元されたデータベースは、バックアップが行われた時と同じ状態になります。

ストレージ製品の高速コピー機能により取得した分割ミラーをコピー・バックすることにより、RESTORE コマンドと同様の機能を実行することもできます。ただし、この後 ROLLFORWARD コマンドによりロールフォワード保留状態を解除する必要があります。

1-2-1-3 ロールフォワード・リカバリー

ログを ROLLFORWARD コマンドで適用し、ある時点までの状態に戻す回復です。保存ログ方式であることが前提条件です。データベースのバックアップ・イメージの RESTORE 後、あるいは、データベースや表スペースがロールフォワード保留状態になった際に使います。

1-2-2 必要な許可

	sysadm	**sysctrl**	**sysmaint**
BACKUP コマンド	○	○	○
既存のデータベースへの RESTORE コマンド	○	○	○
新規のデータベースへの RESTORE コマンド	○	○	－
ROLLFORWARD コマンド	○	○	○

1-2-3 回復活動の記録

DB2 では回復活動記録ファイルに、主に回復関連の活動を記録しています。例えば以下のコマンド操作が記録されます。

BACKUP DATABASE / RESTORE DATABASE / ROLLFORWARD DATABASE / LOAD / QUIESCE TABLESPACE / ALTER TABLESPACE / RENAME TABLESPACE / DROP TABLE / REORGANIZE TABLE

回復活動記録ファイルは LIST HISTORY コマンドで内容表示できます。

このファイルは記録保存が主目的ですので、回復時の RESTORE や ROLLFORWAR コマンドで直接使用されるわけではありませんが、増分バックアップの回復順番を判断する際（RESTORE の AUTOMATIC オプション使用時、あるいは、db2ckrst コマンド使用時）には使用されます。

1-2-4 DB2 システムによる回復対象外のもの

以下の情報は上記の回復方法の対象外となるため、必要に応じて別の方法で回復を行う必要があります。

[a] rootvg に作成される情報 → rootvg のバックアップ、あるいは、ファイルそのもののバックアプより回復可能

/etc/services

/etc/rc.db2

/etc/inittab

/var/ifor/nodelock

/var/db2/v81 以下のファイル

[b] インスタンス・ホーム直下のユーザー定義ファイル → ファイルそのもの、設定情報、ソースファイルなどを保存しておけば回復可能

USEREXIT プログラム

ユーザー定義関数

ストアド・プロシージャー

db2profile

db2systm(データベース・マネージャー構成パラメーター)

レジストリー変数 (profile.env)

db2nodes.cfg

[c] ディレクトリー情報 → 実行済みの CATALOG コマンドの保存、あるは、LIST xx DIRECTORY コマンドの出力を保存しておけば回復可能

データベース・ディレクトリー

ノード・ディレクトリー

DCS ディレクトリー

1-3 バックアップ関連コマンド使用例

データベースのバックアップは DB2 の BACKUP コマンド、またはストレージ製品の高速コピー機能を使用します。

1-3-1 オフライン・バックアップ

【説明】

バックアップ以外のタスクがデータベースに接続しているとオフライン・バックアップが取得できませんので、接続がないことを事前に確認してください。

【使用例】

```
$ db2 list applications
許可 ID  アプリケーション名 アプリケーション・ハンドル  アプリケーション ID    DB 名   エージェント数
-------- ------------- ---------- ------------------------------ -------- -----
DB2INST1 db2bp             31           *LOCAL.db2inst1.080BF4060451    SAMPLE    1
$ db2 force application all
  DB20000I  FORCE APPLICATION コマンドが正常に終了しました。
  DB21024I  このコマンドは非同期であり、即時に有効にならない 場合もあります。
$ db2 deactivate database < database-alias >
```

※データベース起動時に ACTIVATE DATABASE していた場合に必要。

```
  DB20000I  DEACTIVATE DATABASE コマンドが正常に終了しました。
$ db2 backup database <database-alias> user <user-name> using <password> to
```

〈*backup-directory*〉

バックアップは成功しました。このバックアップ・イメージのタイム・スタンプは 20020502162445 です。

1-3-2 オンライン・バックアップ

【説明】

BACKUPコマンドにonline オプションを指定することによりオンライン・バックアップになります。他のアプリケーションやプロセスは接続状態を維持でき表に変更を加えることも出来ます。パックアップ中に行われた変更内容はログ・ファイルに保存されるため、保存ログ方式であることが前提であり、また、回復時にはオンライン・バックアップ開始時点から終了時点までのログが必ず必要になります。

【使用例】

$ db2 backup database 〈*database-alias*〉 user 〈*user-name*〉 using 〈*password*〉 online to 〈*backup-directory*〉

バックアップは成功しました。このバックアップ・イメージのタイム・スタンプは 20020628184432 です。

※回復活動記録ファイルより、最低限必要なログ・ファイルが判別可能。下記例ではS S0000000.LOGとS0000001.LOG が最低限必要なログ・ファイルです。

$ db2 list history backup all for 〈database-alias〉

オプション オブジェクト タイムスタンプ+シーケンス タイプ Dev 最初のログ 現在のログ バックアップ ID

```
-- --- ------------------ ---- --- ------------ ------------ --------------
 B D  20020628184432001  N    D  S0000000.LOG S0000001.LOG  20020627162432001
-------------------------------------------------------------------------------
 2 個の表スペースを含みます :
 00001 SYSCATSPACE
 00002 USERSPACE1
-------------------------------------------------------------------------------
 コメント: DB2 BACKUP 〈database-alias〉 ONLINE
 開始時刻: 20020628184432
 終了時刻: 20020628184458
-------------------------------------------------------------------------------
 00002 ロケーション: /backup/db
```

1-3-3 増分バックアップ

【説明】

BACKUP コマンドに incremental オプションを指定することにより増分バックアップになります。(デルタ方式の場合はさらに delta オプションを指定します。) 増分バックアップはオンライン、オフライン・バックアップ、及びデータベース、表スペース各レベルのバックアップをサポートします。 なお、増分バックアップを用いて回復する際には、必ずそれ以前のフル・バックアップもあわせて必要です。

【使用例】

① TRACKMOD データベース構成パラメーターの確認

$ db2 get db cfg for 〈*database-alias*〉

```
   Track modified pages                                (TRACKMOD) = ON
```

※ (TRACKMOD) = OFF の場合は下記コマンドでデータベース構成パラメーターを変更してください。

$ db2 “update db cfg for 〈*database-alias*〉 using TRACKMOD ON”

(データベースへの最初の接続が完了した時点で変更が有効になります。)

※TRACKMOD 構成パラメーターの変更は必ずデータベース全体のバックアップを取得する前に行って下さい。

②データベース全体のフル・オフライン・バックアップを取得

$ db2 backup database 〈*database-alias*〉 user 〈*user-name*〉 using 〈*password*〉 to〈*backup-directory*〉

バックアップは成功しました。このバックアップ・イメージのタイム・スタンプは 20020502162445 です。

※フルバックアップ後、表スペースを新規作成した場合、その表スペースのフルバックアップを再度取らないとそのあとの増分バックアップができませんのでご注意ください。

③データベースの更新処理後、増分オフライン・バックアップを取得

$ db2 backup database 〈*database-alias*〉 user 〈*user-name*〉 using 〈*password*〉 incremental delta to 〈*backup-directory*〉

バックアップは成功しました。このバックアップ・イメージのタイム・スタンプは 20020502162454 です。

1-3-4 表スペースのバックアップ方法

【説明】

BACKUPコマンドにtablespace オプションと、対象とする表スペース名(一つまたは複数)を指定することにより表スペース・バックアップになります。表スペース・レベル・バックアップは、特定の表スペースのみに影響する問題を回復するときに使用します。表スペース・レベル・バックアップは、オンラインでもオフラインでも行えますが、保存ログ方式であることが前提であり、また、回復時には必ずログが必要になります。

【使用例】

表スペースのオフライン・バックアップを取得します

$ db2 “backup database 〈*database-alias*〉 user 〈*user-name*〉 using 〈*password*〉 tablespace 〈*tablespace-name*〉 to 〈*backup-directory*〉

バックアップは成功しました。このバックアップ・イメージのタイム・スタンプは 20020508134903 です。

1-3-5 バックアップ・イメージのテストーdb2ckbkp コマンド

【説明】

db2ckbkpコマンドを用いると、BACKUPコマンドで取得したバックアップ・イメージが復元可能なものか、その保全性を確認ることができます。オプションを指定すれば(-h でオプションが表示されます)バックアップ・ヘッダーに保管されているメタ・データを表示して、特定のバックアップ・イメージについての情報の判別もできます。バックアップが複数ファイルにまたがって取得していた場合には、順序番号 001 より順番に全てのファイル名を羅列してください。

【使用例 】検査したいバックアップ・イメージが 2 ファイルだった場合

$ db2ckbkp 〈*backupfile-name*〉 _ 〈*backupfile-name*〉

[1] Buffers processed: ####

[2] Buffers processed: #

Image Verification Complete - successful.

1-3-6 ストレージ製品の高速コピー機能によるバックアップ

【説明】

ストレージ製品の高速コピー機能を使用して、データベースのバックアップを取得することができます。コピーの前には、SET WRITE SUSPENDコマンドにより、ディスクへの書き込みをサスペンドします。コピーの後には、SET WRITE RESUME コマンドにより、ディスクへの書き込みを再開します。

【使用例】

① データベースに接続します。

```
$ db2 connect to <database-alias>
```

② データベース上で入出力をサスペンドします。

```
$ db2 set write suspend for database
```

③ ストレージ製品の高速コピー機能を使用して、データベースのコピーを取得します。

④ データベース上で入出力を再開します。

```
$ db2 set write resume for database
```

1-4 回復関連コマンド使用例

回復は回復の方法により、RESTART、RESTORE、ROLLFORWARD などのコマンドを組み合わせて使います。

1-4-1 クラッシュ・リカバリー(破損回復)

【説明】

手動実行と自動実行の2種類あります。手動で行う場合は RESTART　DATABASE コマンドを使います。自動実行とは、データベース構成パラメーターの AUTORESTART が ON(デフォルト値)ならば最初のデータベース接続要求で自動的にクラッシュ・リカバリーが実行されるというものです。いずれにしろクラッシュ・リカバリー回復が完了するまでデータベースは使用可能になりませんので、運用形態にあわせてクラッシュ・リカバリー方法を選択してください。

【使用例】データベースのクラッシュ・リカバリーを手動で行う

```
$ db2  restart  database  <database-alias>
 DB20000I  RESTART  DATABASE  コマンドが正常に終了しました。
```

1-4-2 バージョン・リカバリー（復元回復）

1-4-2-1 DB2 の RESTORE コマンドによる回復

【説明】

RESTORE DATABASEコマンドでデータベースのバックアップ・イメージを復元します。ただし、その後ROLLFORWARDコマンドは用いません。必ずしもバックアップを取得したデータベースに対して復元する必要はなく、新規に作成した別のデータベースに復元することも可能です。データベースをRESTORE中はデータベースは使用不可（オフライン）です。表スペース単位の回復は1-4-4-2 節 表スペース単位のロールフォワード・リカバリーを参照ください。バージョン・リカバリーで使えるのはオフラインのデータベース・バックアップのみです。

【使用例】

①データベースをオフラインにします。

$ db2 list application for database <*database-alias*>

SQL1611W　データベース・システム・モニターからデータが 戻されませんでした。

SQLSTATE=00000

※ 該当データベースへの接続があれば、force application コマンドで接続を切断します。

$ db2 deactivate database < *database-alias* > user <*user-name*> using <*password*>

DB20000I　DEACTIVATE DATABASE コマンドが正常に終了しました。

②データベース全体の復元を実行します。

$ db2 restore db < *database-alias* > user < *user-name* > using < *password* > from <*backup-directory*> without rolloing forward

SQL2539W　警告 ！　バックアップ・イメージ・データベースと同じ既存データベース を復元します。データベース・ファイルは削除されます。

続けますか。(y/n) y

DB20000I　RESTORE DATABASE コマンドが正常に終了しました。

1-4-2-2 ストレージ製品の高速コピー機能による回復

【説明】

ストレージ製品の高速コピー機能を用いてバックアップを取得している場合は、取得したイメージをコピー・バックすることにより回復可能です。

【使用例】

①データベース・インスタンスを停止します。

$ db2stop

① ストレージ製品の高速コピー機能を使用して、1 次システムの分割されたデータをコピー・バックします。分割ログ・ファイルはコピー・バックしないでください。1 次システム上のログはロールフォワード・リカバリーに必要になります。

② データベース・インスタンスを開始します。

$ db2start

③ データベースを初期化します。

$ db2inidb <*database-alias*> as mirror

④ データベースをログの最後まで、またはポイント・イン・タイムまでロールフォワードし、停止します。詳細は1-4-3 ロールフォワード・リカバリーを参照してください。

1-4-3 ロールフォワード・リカバリー

【説明】

保存ログ方式でのみ可能な回復です。 RESTORE 時に without rolling forward キーワードを指定しないとロールフォワード保留状態となるので、ROLLFORWARD コマンドでさらにロールフォワード・リカバリーを行います。また、破損回復後に媒体エラーなどによりデータベースや表スペースがロールフォワード保留状態になることもあります。この時も ROLLFORWARD コマンドでロールフォワード・リカバリーを行います。

ROLLFORWARD コマンドの to end of logs キーワードで最新時点まで、to <*time-stamp*> で指定時点

までロールフォワードします。< *time-stamp* > で指定する時刻は、UTC(世界標準時)がデフォルトになっています。ローカル時間(DB サーバーの時間)を指定する場合は、using local time オプションが必要です。

【使用例】

<使用例 1> データベースを最新時点までロールフォワード・リカバリーします。

① データベースをオフラインにします。

```
$ db2 list application for database <database-alias>
  SQL1611W  データベース・システム・モニターからデータが 戻されませんでした。
  SQLSTATE=00000
```

※ 該当データベースへの接続があれば、force application コマンドで接続を切断します。

```
$ db2 deactivate database < database-alias > user <user-name> using <password>
  DB20000I  DEACTIVATE DATABASE コマンドが正常に終了しました。
```

②データベース全体を復元する。

```
$ db2 restore db < database-alias > user < user-name > using < password > from <backup-directory>
```

③ ロールフォワード状況を確認する。

```
$ db2 rollforward db < database-alias > user < user-name > using < password > query status
                                ロールフォワード状況
      入力データベース別名                 = < database-alias >
      状況を返したノードの数               = 1
      ノード番号                           = 0
      ロールフォワード状況                 = DB  ペンディング
      次に読み込むログ・ファイル           = S0000002.LOG
      処理したログ・ファイル               =  -
      最後にコミットしたトランザクション   = 2002-05-08-04.48.27.000000
```

④ ロールフォワードを実行する。

```
$ db2 rollforward db < database-alias > user < user-name > using < password > to end of logs
                                ロールフォワード状況
      入力データベース別名                 = < database-alias >
      状況を返したノードの数               = 1
      ノード番号                           = 0
      ロールフォワード状況                 = DB  作業中
      次に読み込むログ・ファイル           = S0000004.LOG
      処理したログ・ファイル               = S0000002.LOG - S0000003.LOG
      最後にコミットしたトランザクション   = 2002-05-08-04.49.40.000000
DB20000I  ROLLFORWARD コマンドが正常に終了しました。
```

⑤ ロールフォワードを終了する。

```
$ db2 rollforward db < database-alias > user < user-name > using < password > stop
                                ロールフォワード状況
      入力データベース別名                 = < database-alias >
      状況を返したノードの数               = 1
      ノード番号                           = 0
      ロールフォワード状況                 = 非ペンディング
      次に読み込むログ・ファイル           =
      処理したログ・ファイル               = S0000002.LOG - S0000003.LOG
      最後にコミットしたトランザクション   = 2002-05-08-04.49.40.000000
DB20000I  ROLLFORWARD コマンドが正常に終了しました。
```

<使用例2>データベースを**指定時間まで**ロールフォワード・リカバリーします(POINT IN TIME RECOVERY)。

①データベースをオフラインにします。

```
$ db2 list application for database <database-alias>
SQL1611W  データベース・システム・モニターからデータが 戻されませんでした。
SQLSTATE=00000
```

※ 該当データベースへの接続があれば、force application コマンドで接続を切断します。

```
$ db2 deactivate database < database-alias > user <user-name> using <password>
DB20000I  DEACTIVATE DATABASE コマンドが正常に終了しました
```

②データベース全体を復元する。

```
$ db2 restore db < database-alias > user < user-name > using < password > from <backup-directory>
```

③ロールフォワード状況を確認する。

```
$ db2 rollforward db < database-alias > user < user-name > using < password > query status
                                ロールフォワード状況
      入力データベース別名                 = < database-alias >
      状況を返したノードの数               = 1
      ノード番号                           = 0
      ロールフォワード状況                 = DB  ペンディング
      次に読み込むログ・ファイル           = S0000002.LOG
      処理したログ・ファイル               =  -
      最後にコミットしたトランザクション   = 2002-05-08-04.48.27.000000
```

④ロールフォワードを実行する。

```
$ db2 rollforward db < database-alias > user < user-name > using < password > to <time_stamp>
using local time
                                ロールフォワード状況
      入力データベース別名                 = < database-alias >
      状況を返したノードの数               = 1
      ノード番号                           = 0
      ロールフォワード状況                 = DB  作業中
      次に読み込むログ・ファイル           = S0000004.LOG
      処理したログ・ファイル               = S0000002.LOG - S0000003.LOG
      最後にコミットしたトランザクション   = 2002-05-08-04.49.40.000000
DB20000I  ROLLFORWARD コマンドが正常に終了しました。
```

⑤ロールフォワードを終了する。

```
$ db2 rollforward db < database-alias > user < user-name > using < password > stop
                                ロールフォワード状況
      入力データベース別名                 = < database-alias >
      状況を返したノードの数               = 1
      ノード番号                           = 0
      ロールフォワード状況                 = 非ペンディング
      次に読み込むログ・ファイル           =
      処理したログ・ファイル               = S0000002.LOG - S0000003.LOG
      最後にコミットしたトランザクション   = 2002-05-08-04.49.40.000000
DB20000I  ROLLFORWARD コマンドが正常に終了しました。
```

1-4-4 その他の様々な回復パターン

1-4-4-1 リダイレクト復元

【説明】

RESTORE 実行時に表スペース・コンテナーの追加・変更・削除を行いたい場合の回復方法で、RESTORE コマンドに redirect オプションをつけて行う手順です。

コンテナーが何らかの理由でアクセス不能になっている場合は RESTORE が失敗します。リダイクレト復元を使えば、このような時に別のコンテナーパスを指定する事ができます。

また、SMS 表スペースはコンテナーを増やす事も可能です。 ただし、DMS 表スペース ⇔ SMS 表スペースの変更はできません。

【使用例】データベースをリダイレクト復元する。

① redirect オプションを指定して RESTORE DATABASE コマンドを発行する。

```
$ db2 restore database 〈database-alias〉 replace existing redirect
```

※① が正常終了した後で③ が完了する前に、abort オプション付きの RESTORE を発行して復元を打ち切ることができます。

```
$ db2 restore db 〈database-alias〉 abort
```

② 再定義する必要があるコンテナーを持つ表スペースごとに、 SET TABLESPACE CONTAINERS コマンドを発行する。

```
$ db2 set tablespace containers for 〈tablespace-id〉 using 〈container〉
```

※コンテナー指定が成功したかを、 LIST TABLESPACE CONTAINERS コマンドでを確認する。

③ ① および ② が正常終了した後、次のコマンドを発行しリダイレクト復元を完了する。

```
$ db2 restore database 〈database-alias〉 continue
```

※③ が失敗した場合、または復元を打ち切った場合、リダイレクトした復元は① から再始動できます。

1-4-4-2 表スペース単位のロールフォワード・リカバリー

【説明】

表スペース単位での回復も可能であり、データベース単位の場合と同様、ロールフォワード・リカバリーを行うことも可能です。 表スペース単位の回復は、回復操作をオンラインで実行したり(回復対象ではない表スペースへのアクセスは可)、データベース全体のバックアップ・イメージから特定の表スペースのみを回復することも可能です。

【使用例】 特定の表スペースのみ最新時点までロールフォワード・リカバリーします。

①表スペースのみ復元

```
$ db2 “restore database 〈database-alias〉 tablespace 〈 tablespace-name 〉online from 〈backup-directory〉 taken at 〈time-stamp〉 without prompting”
DB20000I RESTORE DATABASE コマンドが正常に終了しました。
```

②RESTORE 終了後該当表スペースはロールフォワード保留状態となり表スペースへのアクセスは SQL0290 エラーとなってしまうのでロールフォワード・リカバリーを行います。

```
$ db2 “rollforward database 〈database-alias〉 to end of logs and stop tablespace (〈tablespace-name〉) online”
```

ロールフォワード状況

```
入力データベース別名                    = < database-alias >
状況を返したノードの数                  = 1

ノード番号                              = 0
ロールフォワード状況                    = 非ペンディング
次に読み込むログ・ファイル              =
処理したログ・ファイル                  = S0000002.LOG - S0000003.LOG
最後にコミットしたトランザクション      = 2002-05-21-04.49.40.000000

DB20000I  ROLLFORWARD コマンドが正常に終了しました。
```

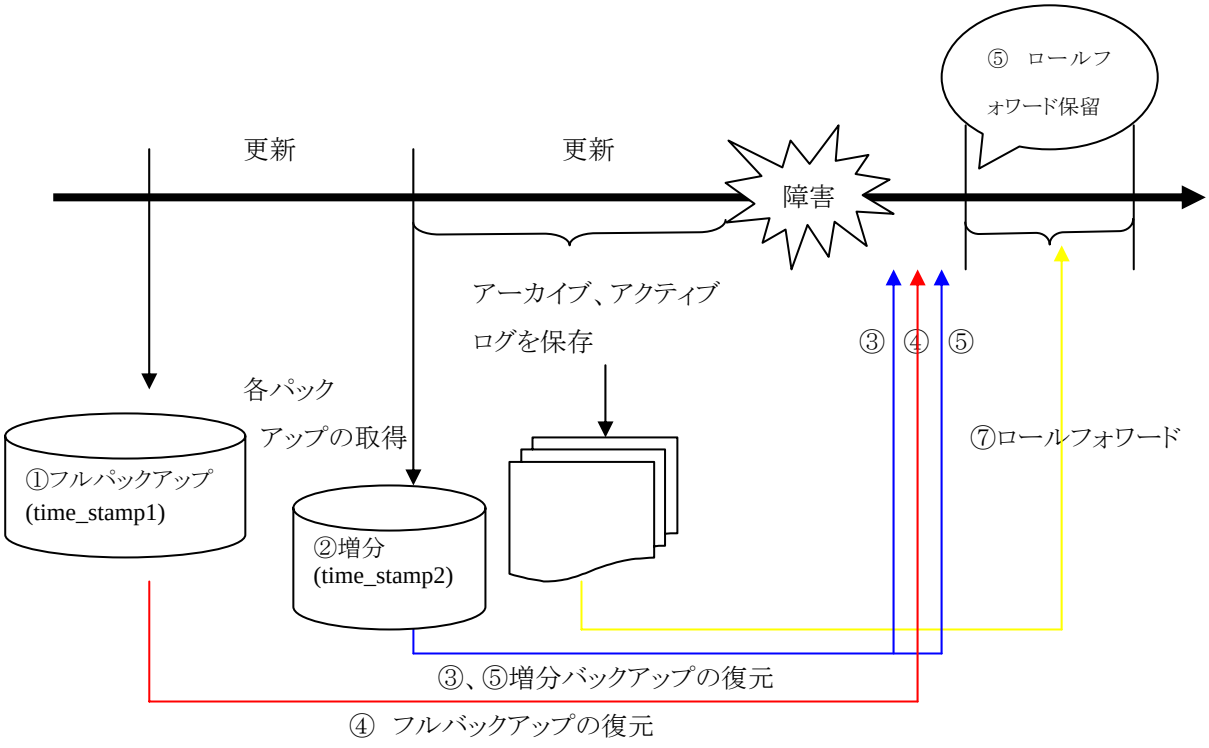
1-4-4-3 増分バックアップからの回復

【説明】

増分バックアップ・イメージからの復元操作の場合、回復ターゲットとなる増分バックアップ・イメージは 2 回アクセスされることになります。最初のアクセスでは増分バックアップ・イメージから初期データだけが読み取られ、ユーザーデータは読み取られません。2 回目のアクセス時には完了イメージのみが読み取られて処理されます。この為、例えばデータベースのフル・バックアップ・イメージが取得後に表の更新処理が行われ、その後何らかの障害が発生した場合に増分バックアップ・イメージからの復元操作を実行すると、初回のアクセス時にはフル・バックアップ取得後に行った更新処理等のユーザーデータは復元されません。増分バックアップ内の更新データを完全に復元するには、もう一度この増分バックアップ・イメージを読み取らなければなりません。

増分バックアップ・イメージを復元するには、RESTORE DATABASE コマンドに incremental オプションと taken at <*time_stamp*>オプションを指定します。復元する最終イメージのタイム・スタンプを指定してください。

<回復シナリオ>



【使用例】

<使用例 1>増分バックアップからデータベース全体を回復します。

① データベースのフルバックアップを取得します。

② データベースの増分バックアップを取得します。

（詳細手順については 1-3-3 節増分バックアップをご覧下さい。）

～ 障害発生 ～

③ ②で保管してあった増分のイメージをまず復元します。

$ db2 “restore db <*database-alias*> incremental from <*backup-directory*>
taken at <*time-stamp2*> without prompting”

SQL2540W 復元は成功しましたが、非割り込みモードでデータベースを復元中に、警告"2539"が出されました。

④ ①で保管してあったフルイメージを復元します。

$ db2 “restore db <*database-alias*> incremental from <*backup-directory*>
taken at < *time-stamp1*> without prompting”

DB20000I RESTORE DATABASE コマンドが正常に終了しました。

⑤ もう一度②で保管してあった増分イメージを復元します。

$ db2 “restore db <*database-alias*> incremental from <*backup-directory*>
taken at < *time-stamp2*> without prompting”

DB20000I RESTORE DATABASE コマンドが正常に終了しました。

⑥ ロールフォワード保留状態になります。

$ db2 connect to <*database-alias*>

SQL1117N 「ROLL-FORWARD PENDING」のために、データベース "<*database-alias*>" の接続または活動化を行うことはできません。

SQLSTATE=57019

⑦ データベース全体のロールフォワードを行います。

$ db2 “rollforward db <*database-alias*> to end of logs and stop”

ロールフォワード状況

入力データベース別名 = < *database-alias* >
状況を返したノードの数 = 1

ノード番号 = 0
ロールフォワード状況 = 非ペンディング
次に読み込むログ・ファイル =
処理したログ・ファイル = S0000002.LOG - S0000003.LOG
最後にコミットしたトランザクション = 2002-06-26-08.11.18.000000

DB20000I ROLLFORWARD コマンドが正常に終了しました。

<使用例 2>AUTOMATIC 指定による増分バックアップからデータベース全体を回復方法

① データベースのフルバックアップを取得します。

② データベースの増分バックアップを取得します。

（詳細手順については 1-3-3 節増分バックアップをご覧下さい。）

～ 障害発生 ～

③ AUTOMATIC 指定による復元作業を行います。

$ db2 “restore db 〈*database-alias*〉 incremental automatic from 〈*backup-directory*〉 taken at 〈*time-stamp2*〉 without prompting”

SQL2540W 復元は成功しましたが、非割り込みモードでデータベースを復元中に、警告"2539"が出されました。

※automatic を指定すると「使用例1の③、④、⑤」の作業が省略されます。

④ ロールフォワード保留状態になります。

$ db2 connect to 〈*database-alias*〉

SQL1117N 「ROLL-FORWARD PENDING」のために、データベース "〈*database-alias*〉" の接続または活動化を行うことはできません。SQLSTATE=57019

⑤ データベース全体のロールフォワードを行います。

$ db2 “rollforward db 〈*database-alias*〉 to end of logs and stop”

ロールフォワード状況

```
入力データベース別名                  = < database-alias >
状況を返したノードの数                  = 1

ノード番号                            = 0
ロールフォワード状況                    = 非ペンディング
次に読み込むログ・ファイル              =
処理したログ・ファイル                  = S0000002.LOG - S0000003.LOG
最後にコミットしたトランザクション          = 2002-06-26-08.11.18.000000
DB20000I  ROLLFORWARD コマンドが正常に終了しました。
```

1-4-4-4 循環ログ方式の場合の一時表スペース破損時の回復

【説明】

一時表スペースに障害が発生するとその表スペースのステータスがOFFLINEになりますが、データベースへの接続は可能です。障害が発生した表スペースが一時表スペースだった場合は再作成すればよいので、データベースへの接続後、新規の一時表スペースを作成し、壊れた一時表スペースを削除する方法が可能です。(2 章 7-1-2 節一時表スペースの破損参照)

また循環ログ方式の場合に限り、データベース接続時にクラッシュ・リカバリー回復が必要で、且つ一時表スペースに障害が発生している場合には、まず RESTART コマンドに、DROP PENDING TABLESPACEオプションを指定すればクラッシュ・リカバリーを正常終了させデータベースに接続することができるようになります。

【使用例】 データベース接続時に破損回復が必要で、且つ一時表スペースに障害が発生している場合

（障害時の db2diag.log 出力例）

```
2002-07-04-16.42.09.879052   Instance:inst   Node:000
PID:163014(db2agent (SAMPLE))   Appid:*LOCAL. inst.020704073921
base_sys_utilities  sqledint   Probe:0   Database:SAMPLE
Crash Recovery is needed.

2002-07-04-16.42.10.093265   Instance:inst   Node:000
PID:163014(db2agent (SAMPLE))   Appid:*LOCAL.inst.020704073921
buffer_pool_services  sqlbStartPools   Probe:3   Database:SAMPLE
```

```
TEMPSPACE1

2002-07-04-16.42.10.242488   Instance:inst   Node:000
PID:163014(db2agent (SAMPLE))   Appid:*LOCAL.inst.020704073921
buffer_pool_services  sqlbSMSDoContainerOp   Probe:815   Database:SAMPLE
DIA9999E An internal error occurred. Report the following error code :
"0xFFFFC11E".

2002-07-04-16.42.10.278705   Instance:inst   Node:000
PID:163014(db2agent (SAMPLE))   Appid:*LOCAL.inst.020704073921
buffer_pool_services  sqlbSMSDoContainerOp   Probe:815   Database:SAMPLE
Error checking container 0 (/database/NODE0000/SQL00001/SQLT0001.0) for tbsp 1.
 Rc = FFFFF611

2002-07-04-16.42.10.411960   Instance:inst   Node:000
PID:163014(db2agent (SAMPLE))   Appid:*LOCAL.inst.020704073921
buffer_pool_services  sqlbStartPools   Probe:30   Database:SAMPLE
Tablespace TEMPSPACE1 (1) is in state x0.
Starting it failed. rc=ffff c11e                                    ..ﾁ.
```

① RESTART コマンドで DROP PENDING TABLESPACE オプションを指定して実行します。

$ db2 “restart database <database-alias> drop pending tablespaces <*tablespace-name*>”

DB20000I RESTART DATABASE コマンドが正常に終了しました。

② データベースに接続します。

$ db2 connect to <database-alias>

③ 新規の一時表スペースを作成します。

$ db2 “create system temporary tablespace <*tablespace-name*> managerd by system using (‘<*container string*>’)”

DB20000I SQL コマンドが正常に終了しました。

④ 壊れた一時表スペースを削除します。

$ db2 “drop tablespace <*tablespace-name*>”

DB20000I SQL コマンドが正常に終了しました。

1-4-4-5 除去された表の回復(Dropped Table Recovery)

【説明】

dropped table recovery on オプション付きの表スペース(CREATE TABLESPACE / ALTER TABLESPACE で指定) 内に作成した表は、誤って表を除去してしまった場合でも回復させる事ができる、という回復方法です。

1回の回復手順(下記使用例を参照)で一つの表のみ回復できます。そのため、複数の表を誤って除去した場合は回復手順を複数回繰り返す必要があります。

【使用例】表スペースに Dropped Table Recovery の設定をし、表を除去してしまった場合の回復

① 表スペース作成時に dropped table recovery on の指定をする。

$ db2 “create tablespace < *tablespace-name*>

managed by database using (‘*<container string>*’) dropped table recovery on”

DB20000I SQL コマンドが正常に終了しました。

② データベースに接続します。

$ db2 “create table *<table-name>* (col1 integer , col2 integer) in *< tablespace-name>*

DB20000I SQL コマンドが正常に終了しました。

③ 表の作成。

$ db2 “create table *<table-name>* (col1 integer , col2 integer) in *< tablespace-name>*

DB20000I SQL コマンドが正常に終了しました。

～ 誤って表を除去したと仮定 ～

④ 回復活動記録ファイルの表 ID(Backup ID) と DDL を参照しておく。

$ db2 list history dropped table all for *<database-alias>*

*<database-alias>*の履歴ファイルのリスト

突きあわせファイル項目数＝1

```
Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log Backup ID
--- ----- -------------------------- ------ ----- -------------- -------------- --------- ---
D  T  20020521201601                                          <table_id>
------------------------------------------------------------------------------------------------
  “< table-name >”は 1 表スペースに常駐します :
 00001  < tablespace-name >
------------------------------------------------------------------------------------------------
   コメント : DROP TABLE
開始時間 : 20020521201601
  終了時間 : 20020521201601
------------------------------------------------------------------------------------------------
 00001
       DDL : CREATE TABLE “<table-name >” (“C1” INTEGER , “C2” INTEGER)
                                                                                          in
<tablespace-name>
```

⑤ 復元します。

$ db2 “restore db < *database-alias* > user < *user-name* > using < *password* >

from <*backup-directory*>”

DB20000I RESTORE DATABASE コマンドが正常に終了しました。

⑥ ROLLFORWARD に recover dropped table オプションを追加する。<*table_id*> は③で得られた情報より対象表の表 ID を指定する。<*export-directory*>は除去してしまった表のデータを EXPORT するパス名で、/<*export-directory*>/NODEnnnn/data（nnnn はノード番号なので、常に 0000）というファイル名で作成される。

$ db2 “rollforward database *<database-alias>* recover dropped table *<table_id>*

to *<export-directory>*”

ロールフォワード状況

入力データベース別名 = *<database-alias>*
状況を返したノードの数 = 1

```
ノード番号                                 = 0
ロールフォワード状況                       = 非ペンディング
次に読み込むログ・ファイル                 =
処理したログ・ファイル                     = S0000002.LOG - S0000003.LOG
最後にコミットしたトランザクション         = 2002-05-21-04.49.40.000000
DB20000I  ROLLFORWARD コマンドが正常に終了しました。
```

⑦ データベースに接続します。

```
$ db2 connect to <database-alias>
```

⑧ ④で得た DDL を元に、表を再作成する。

```
$ db2  “create  table  <table-name> (c1 integer , c2 integer)  in  < tablespace-name >
DB20000I SQL コマンドが正常に終了しました。
```

⑨ ロールフォワード中に除去された表のデータが EXPORT されているので、それを IMPORT あるいは LOAD する。

```
$ db2  “import from /<export-directory>/NODEnnnn/data of del replace into <ablespace-name>”
SQL3109N ユーティリティーが，ファイル ”/<export-directory>/NODEnnnn/data ” からデータのロードを開始しています。
SQL3110N ユーティリティーが処理を完了しました。”n”行が入力ファイルから読み取られました。
SQL3221W ...COMMIT WORK が開始されました。入力レコード数 = ”n”。
SQL3222W ...すべてのデータベース変更のコミットが成功しました。
SQL3149N ”n”行が、入力ファイルから処理されました。 ”n” 行が，正常に表に挿入されました。
0 行が，拒否されました。

読み込まれた行数 = n
スキップされた行数 = 0
挿入された行数 = n
更新された行数 = 0
拒否された行数 = 0
コミットされた行数 = n
```

⑧ データは回復されているが、表自体は新たに作成された形になるので、索引の作成、RUNSTATS の実行、表を用いるアプリケーションの BIND など、必要に応じて行う。

2 エラー情報の記録

2-1 DB2 のログ情報

2-1-1 診断ログ(db2diag.log)

【説明】

DB2 のエラー、警告、通知メッセージなどが記録されています。何か障害が発生した際には、まず参照すべきファイルです。障害発生時間の特定を行い、該当時間にエラーが出力されているか調査を行ってください。

出力されるログ内容は診断情報のレベルによって内容が異なります。

なお、DB2 は db2diag.log の内容を自動的には削除しないため、大きくなる一方です。大きくなり過ぎた場合は必要に応じてファイルをバックアップ後、消去してください。消去してもDB2が db2diag.log を必要とする時に自動的に生成されます。

<情報の出力レベル>

診断情報の出力レベルは以下の4種類あります。

DIAGLEVEL(0) - 診断データは取り込みません。

DIAGLEVEL(1) - 重大エラーのみ

DIAGLEVEL(2) - 全てのエラー(重大エラー及び非重大エラー)

DIAGLEVEL(3) - 全てのエラー及び警告

DIAGLEVEL(4) - 全てのエラー、警告、通知メッセージ、及びその他の内部診断情報

※通常(デフォルト)AIX に於いての出力レベルは3であるが、障害の特定、再現性がある場合には詳細なエラー内容を取得する為に、出力レベルを4に上げログの取得を行う。

診断情報のレベル変更は、データベースマネージャー構成パラメーターの DIAGLEVEL を変更し、DB2 を再起動します。

例:出力レベルを 4 に変更

```
$ db2   "update   dbm   cfg   using   diaglevel   4"
```

その後 DB2 再起動

<情報の出力先>

診断情報の出力先のデフォルトは $INSTHOME/sqllib/db2dump です。

診断情報の出力変更は、データベースマネージャー構成パラメーターの DIAGPATH を変更し、DB2 を再起動します。

例:診断パスをデフォルト $INSTHOME/sqllib/db2dump から /test/db2dump へ変更

```
$ db2 "update   dbm   cfg   using   DIAGPATH   /test/db2dump"
```

その後 DB2 再起動

【解釈の例】

詳細は『問題判別の手引き』を参照して下さい。

<db2diag.log のヘッダー情報>

```
1997-03-16-11.53.18.001160 ①    Instance:payroll ②  Node:000 ③
PID:44829(db2agent (SAMPLE)) ④     Appid:*LOCAL.payroll.970317140834 ⑤
lock_manager ⑥          sqlplrq ⑦    Probe:111 ⑧  Database:SAMPLE ⑨
DIA9999E ⑩ An internal return code occurred. Report the following:
"0xFFFFE10E". ⑪
```

凡例:

① メッセージのタイム・スタンプ。

② メッセージを生成するインスタンス名。

③ DB2 エンタープライズ拡張エディション・システムで db2nodes.cfg ファイルを使用する場合、メッセージを生成するノード。 (エンタープライズ・エディションでは、値は "000" です。)

④ メッセージを生成するプロセスの識別。この例では、メッセージは 44829 と識別されるプロセスから送られています。このプロセス名は db2agent で、SAMPLE という名前のデータベースに接続されています。

※ アプリケーションが DUOW (分散作業単位)環境で実行されている場合、表示されている ID は DUOW 相関トークンです。

⑤ プロセスのアプリケーションの ID。この例では、メッセージを生成しているプロセスは、*LOCAL.payroll.970317140834 という ID のアプリケーションのために作業しています。アプリケーション ID より、payroll というインスタンスへのローカル接続であることがわかります。仮に TCP/IP 接続のアプリケーションだった場合は、例えば「0915106D.0805.980303200328」のように表示され、IP アドレスもわかります。例では 0915106D より IP アドレス 9.21.16.109 からのリモート接続であることがわかります。

⑥ メッセージを書き込む DB2 構成要素。

⑦ メッセージを提供する関数名。この関数は、メッセージを書き込む DB2 副構成要素内で処理されます。関数が実行する活動の種類について詳しく知るためには、名前の 4 番目の文字を見ます。この例では、関数 "sqlplrq" の文字 "p" は、データ保護の問題であることを表します (たとえば、ログの損傷など) 。 次のリストに、関数名の 4 番目に使用される文字、およびその文字によって表される活動の種類を示します。

b バッファー・プール

c クライアント / サーバー間の通信

d データ・マネージメント

e エンジン処理

o オペレーティング・システムの呼び出し (ファイルのオープンおよびクローズ)

p データ保護 (ロックおよびロギング)

r 関係データベース・サービス

s ソート

x 索引付け

⑧ 報告された内部エラーの識別。

⑨ エラーが発生したデータベース。

⑩ 内部エラーが生じたことを示す診断メッセージ。

⑪ 内部戻りコードの 16 進表示。詳細については、<16 進コードの解釈>を参照してください。

<SQLCA 構造>

重大エラーの場合、SQLCA 構造は db2diag.log ファイルにダンプされます。各 SQLCA フィールドの詳細については、『問題判別の手引き 付録 C, SQL 連絡域 (SQLCA) 』を参照してください。

＊SQLCA ダンプがある db2diag.log ファイルの例を次に示します。

```
1997-03-16-11.53.18.001160     Instance:payroll    Node:000
PID:44829(db2agent (SAMPLE))      Appid:*LOCAL.payroll.970317140834
relation_data_serv           sqlrerlg     Probe:17    Database:SAMPLE
DIA9999E An internal return code occurred. Report the following : "0xFFFFE101".
Data Title :SQLCA pid(14358) ①
sqlcaid : SQLCA    sqlcabc: 136    sqlcode: -980 ②  sqlerrml: 0
sqlerrmc: ③
 sqlerrp : sqlrita
sqlerrd ④: (1) 0xFFFFE101  ⑤  (2) 0x00000000     (3) 0x00000000
          (4) 0x00000000     (5) 0x00000000     (6) 0x00000000
sqlwarn : (1)          (2)         (3)         (4)         (5)         (6)
        (7)          (8)         (9)        (10)        (11)

sqlstate:
```

凡例:

① SQLCA 項目の始め。

② SQL 状態 (負の数の場合、エラーが生じています)。

③ SQL エラー・コードに関連した理由コード。

④ 最終 SQL エラー・コードの原因となったエラーがいくつかある場合があります。これらのエラーは sqlerrd 領域に順番に表示されます。

⑤ SQL エラーの 16 進表示。詳細については、<16 進コードの解釈>を参照してください。

＜16 進コードの解釈＞

AIX ではエラー・コードが ffff *nnnn* 形式で出力されるため、*nnnn* を以下のように解釈します。

1. 16変換ツールを使用して、10 進形式に変換します。そのコードが『*メッセージ解説書* 』にある場合、それは SQL コードです。
2. エラー・コードの 10 進変換が SQL コードではない場合、戻りコードです。戻りコードのリストは、『問題判別の手引きDB2 内部戻りコード』を参照してください。

OS/2 および Windows システムでは、db2diag.log ファイル、または SQLCA エラー・コードのバイトが逆転しているものがあります。形式が ffff *nnnn* の場合、そのまま解釈されます。形式が *nnnn* ffff の場合、意味を読み取るためには、バイトを逆転しなければなりません。

そのためには、最初の 4 文字を最後の 4 文字と入れ替え、次に、5 番目と 6 番目の文字を、7 番

目と 8 番目の文字と入れ替えます。 たとえば、エラー・コード "0ae6 ffff" は "ffff e60a" と変換されます。

2-1-2 アラート・ログ(AIX:syslog)

【説明】

DB2 のアラート・ログ・ファイルは、UNIX システムでは syslog ファイルに出力されます。前述のdb2diag.log のみ使用することが多いようですが、syslog にも db2diag.log と同様の有益な情報があることがあります。 使うためにはユーザーは syslog を自分で設定する必要があります。

【設定方法】

/etc/syslog.conf を編集します。機能名、エラーレベル、メッセージの宛先などを設定します。 機能名は user を設定します。エラーレベルは以下より選択可能です。宛先のファイルの世代管理もあわせて検討してください。

<使用可能なエラーレベル>

alert:アラートメッセージのみが記録されます。

err :エラー及びアラートメッセージが記録されます。

warn :警告、エラー、及びアラートメッセージが記録されます。

info :情報、警告、エラー、及びアラートメッセージが記録されます。

【使用例】warn レベルを /tmp/syslog.dat に出力します。

① /syslog/conf に以下の内容を追加

user.warn /tmp/syslog.dat

②あらかじめ出力宛先ファイルは作成

$ touch /tmp/syslog.dat

③ syslog デーモンをリフレッシュ

#refresh -s syslogd

2-1-3 DB2ダンプ・ファイル/DB2 コアファイル

【説明】

エラーが発生すると、問題を診断する際に役立つと思われる追加情報(内部制御ブロックなど)がある場合に、ダンプ・ファイルが作成されます。ダンプ・ファイルに書き込まれる全てのデータ項目には、関連したタイム・スタンプがあり、問題判別に役立ちます。ダンプ・ファイルはバイナリ形式です。サポートセンターからこれらのファイルの提出を依頼される場合がありますので、保管をお願いいたします。

ダンプ・ファイルが作成または追加される場合、時刻および書き込まれたデータのタイプを示す項目がdb2diag.log に追加されます。これらの db2diag.log 項目は次のようなものです。

```
------------------------------------------------------------------------------------------------
---------------
2002-06-13-21.08.48.300950   Instance:db2inst1   Node:000
PID:118328(db2agent (TESTDB))   Appid:*LOCAL.db2inst1.020613120344
relation_data_serv  sqlrr_signal_handler   Probe:10   Database:TESTDB
DIA7107I Execution of a component signal handling function has begun.
```

```
Dump File /home/db2/sqllib/db2dump/56772.000  Data:Recursive dump_sect
--------------------------------------------------------------------------------------------------
---------------
```

この例では、/home/db2/sqllib/db2dump/56772.000 という名前のファイルがダンプ・ファイルに該当します。

UNIX ベースのシステムでは、ダンプ・ファイルはメモリー・ダンプ・ディレクトリーに作成される場合があります。これらのファイルは DB2 コアファイルと呼ばれ、DB2 固有のものです。db2diag.log に記録されているダンプ・ファイルに加え、これらの DB2 コアファイルも収集してください。

DB2 コアファイルはデータベースマネージャー構成パラメーターの DIAGPATH のコア・ディレクトリーに出力されることが多いです。コア・ディレクトリーは各プロセスごとに 1 つあります。ディレクトリー名は文字”c”で始まり、影響を受けたプロセスの ID(pid)番号が続きます。

例えば、$INSTHOME/db2dump /c63422.000 は pid 63422 のプロセスのコアファイルを含むディレクトリーです。

【確認手順】

①db2diag.log によりダンプ・ファイルの出力確認

```
$ view <diag_path>/db2diag.log
```

ダンプ・ファイル出力表示があれば、サポートセンター送付用に該当ファイルを保管。

②DB2 コアファイルの検索とその作成日付の確認

```
$ find  <diag_path> -name c*.000 -print
$ find  $INSTHOME/. -name c*.000 -print
```

DB2 コアファイルの出力があれば、サポートセンター送付用に該当ファイルを保管。

また、DB2 コアファイルはシステムコアファイルと同様解析可能です。手順は 2-3-3 節<u>core</u>を参照してください。

2-1-4 DB2 トラップ・ファイル

【説明】

トラップやセグメンテーション違反、例外などにより処理が続行できないときに生成され、問題発生前に実行されていた最後のステップの関数の流れが含まれています。

DB2 によって発行されたすべてのシグナルおよび例外は DB2 トラップ・ファイルに報告されます。トラップ・ファイルには、システム停止前に実行された最後のステップの関数の流れが含まれています。サポートセンターからこれらのファイルの提出を依頼される場合がありますので、保管をお願いいたします。

DB2 トラップ・ファイルはデータベースマネージャー構成パラメーターの DIAGPATH で指定されたパスに出力されます。

ファイル名の最初の文字は”t”で、その後にプロセス ID(pid)が続きます。

例えば、t62122.000 はプロセス ID62122 のプロセスのトラップ・ファイルです。

【確認手順】

①データベースマネージャー構成パラメーターの DIAGPATH を確認します。

```
$ db2 get dbm cfg | grep DIAGPATH
```

診断データのディレクトリー・パス　　(DIAGPATH) = <*diag_path*>

②DB2トラップ・ファイルの存在とその作成日付を確認します。

```
$ cd <diag_path>
$ ls -l
```

2-2 DB2 のトレース

2-2-1 DB2トレース(db2trc)

【説明】

db2trc コマンドにより、トレース取得のほか、可読形式へのフォーマットもできます。必ずインスタンスオーナーで実行します。 問題判別の為にトレースを取得する為には、トレースをオンにしている間にエラーを再現しなければなりません。ただし、取得タイミングに関連して問題が再現しないことがありますので注意が必要です。トレースで収集された情報量は非常に大きくなり、パフォーマンスも極端に低下しますので、必要な時のみ、また、極力トレースでとりたい活動以外は起こさないような状態で取得します。またトレースバッファーファイルのサイズが指定の値を超えると、最初の情報から上書きされますので、十分な大きさのファイルを設定することをお奨めいたします。

【使用例】

<使用例1>　メモリーを使用したトレース

① トレースをオンにします。(トレースバッファーファイル 5MB)

```
$ db2trc on -l 5000000 -e -1
```

② 問題のシナリオの再現を行います。

③ トレースをファイルにダンプします。

```
$ db2trc dump <trace-dump-file>
```

④ トレースをオフにします。

```
$ db2trc off
```

⑤ トレースを可読形式にします。

```
$ db2trc flw <trace-dump-file> <trace-flw-file>
```
※ プロセスまたはスレッドでのソート形式

```
$ db2trc fmt <trace-dump-file> <trace-fmt-file>
```
※ 全イベントの年代順でのリスト形式

※ トレースをフォーマットすることでトレースが wrap しているかどうか確認できます。

Trace wrapped　　: NO <==== NO であることの確認。

YES の場合は wrap してしまっているので①で指定する-l の後のサイズを増やして再度やりなおしてください。

⑥ サポートセンターに送付する際にはで取得したバイナリーファイル (<*trace-dump-file*>)、⑤でflw および fmt でフォーマットしたファイル(<*trace-flw-file*> <*trace-fmt-file*>)の3つを送付します。

※もしトレースオンの状態でDB2が異常終了した場合、トレースは自動的に停止されます。明示的に停止する必要はありません。トレース開始から異常終了までの間に取得されたデータは、DIAGPATH、あるいは $INSTHOME/sqllib/db2dump に「db2trdmp.ノード番号」の形式で自動的に出力されます。

<使用例 2>　ファイルを使用したトレース

システムがハングするような障害で、メモリーに取得したトレースをダンプできない際に使います。メモリーを使用したトレースよりさらにパフォーマンスの悪化を引き起こします。

①トレースをオンにします。(トレースバッファーファイル 5MB)

```
$ db2trc on -l 5000000 -e -1 -f db2trc.dmp
```

②問題のシナリオの再現を行います。

③システムをリブートすると db2trc.dmp というファイルができあがります。

後は使用例1と同様にフォーマットします。

出力例については『問題判別の手引き DB2トレース機能の使用』を参照してください。

2-2-2 CLI(ODBC)/JDBC トレース

【説明】

DB2 の CLI ドライバーーと通して実行される CLI/ODBC/JDBC/SQLJ を使用したアプリケーションをトレースする機能です。CLIトレースより、実際にアプリケーションよりどのような命令を受けているかが分かります。また、CLI のパラメーター設定が実際のアプリケーション接続に正しく反映されているかを確認するのにも使えます。JDBCアプリケーションの場合は、JDBCトレースも併せて取得する必要があります。なお、アプリケーションを実行しているクライアント上で取得することにご注意ください。

なお、V8.1 より提供が開始された、Universal JDBC Driver の場合には、多くの部分で、CLI を使用しないようになっていますので、CLI トレースから十分な情報を取得することができません。

【取得するための設定】

① クライアントのdb2cli.iniファイルを編集してください。

そのために、以下のようにコマンドを発行してください。

```
$ db2 update cli cfg for section common using trace 1
```

※ db2cli.iniファイルは AIXならば $INSTHOME/sqllib/cfgの下に、WindowsNT/2000ならば DB2の導入ディレクトリー(x:SQLLIB)の下にあります。

※ [COMMON] セクションに以下の行を追加します。

・「trace」で始まる5行がCLIトレース、「jdbc」で始まる4行がJDBCトレースの設定です

・ 結果ファイルは、ここで指定した場所にスレッド単位で作成されます。ディレクトリーはあらかじめ作成しておきます。(CLIトレースとJDBCトレースは異なる出力パスを指定した方がよいでしょう。)

<===上の[]の終わりと[COMMON]のはじまりの間に1行改行が必要

```
[COMMON]
trace=1
tracepathname=/xx/clitracepath
traceflush=1
tracecomm=1
tracetimestamp=1
jdbctrace=1
jdbctracepathname=/xx/jdbctracepath
jdbctraceflush=1
jdbctracecomm=1
```

<===ファイルの終わりに1行改行が必要

～ 以下のセクション省略 ～

② db2 に設定した各種パラメーターを確認してください。

```
$ db2 get cli cfg for section common
                セクション: COMMON
-------------------------------------------------
          trace=1
          tracepathname=/xx/clitracepath
          traceflush=1
          tracecomm=1
          tracetimestamp=1
          jdbctrace=1
          jdbctracepathname=/xx/jdbctracepath
          jdbctraceflush=1
          jdbctracecomm=1
```

③ アプリケーションを再始動してください。

④ トレース取得が終了したら、①で編集したdb2cli.iniファイルを元の設定に戻し、アプリケーションを再始動してください。　（トレースが取得され続けてしまうため。）

【使用例】

db2cli.ini に記述したパラメーターがアプリケーションに正しく反映されているか確認します。

(Windows2000 クライアントの例)

① db2cli.ini を以下のように設定してあり、SAMPLE データベースへの接続に “PATCH1=1024”が反映されているかを確認したいとします。

※ ただし、[COMMON]セクションではなく、[<DATABASE-ALIAS>]セクションに記述しないとトレース上表示されません。

```
[COMMON]
trace=1
tracepathname=c:¥temp1
traceflush=1
tracecomm=1
tracetimestamp=1
jdbctrace=1
jdbctracepathname=c:¥temp2
jdbctraceflush=1
jdbctracecomm=1

[SAMPLE]
PATCH2=6
PATCH1=1024
LOBMAXCOLUMNSIZE=1048575
LONGDATACOMPAT=1
DBALIAS=SAMPLE
```

② アプリケーションを実行してください。

③ SAMPLE データベースへの接続に “PATCH1=1024”が反映されているか作成されたトレースより確認してください。

※ 出力されたトレースファイルの中でSQLDriverConnect（あるいは同種のAPI)の Response の出力部分に記述されます。

```
[1025232904.018762 - 06/28/2002 11:55:04.018762] SQLDriverConnect( hDbc=0:1, hwnd=6:1296,
```

```
szConnStrIn="DSN=SAMPLE;REPAINT=2",          cbConnStrIn=-3,          szConnStrOut=&01dde470,
cbConnStrOutMax=256, pcbConnStrOut=&01dde344, fDriverCompletion=SQL_DRIVER_PROMPT )
[1025232904.196813 - 06/28/2002 11:55:04.196813]       ---> Time elapsed - +1.364000E-003
seconds
    sqlccsend( ulBytes - 1616 )
    sqlccsend( Handle - 20101360 )
    sqlccsend( ) - rc - 0, time elapsed - +1.598000E-003
    sqlccrecv( )
    sqlccrecv( ulBytes - 1262 ) - rc - 0, time elapsed - +1.072000E-003
    sqlccsend( ulBytes - 603 )
    sqlccsend( Handle - 20101360 )
    sqlccsend( ) - rc - 0, time elapsed - +1.417000E-003
    sqlccrecv( )
    sqlccrecv( ulBytes - 237 ) - rc - 0, time elapsed - +2.381480E-001
[1025232907.490710 - 06/28/2002 11:55:07.490710] ( DBMS NAME="DB2/NT", Version="07.02.0002",
Fixpack="0x23030105" )
[1025232907.493578 - 06/28/2002 11:55:07.493578]
[1025232907.494282 - 06/28/2002 11:55:07.494282] ( Application Codepage=943, Database
Codepage=943, Char Send/Recv Codepage=943, Graphic Send/Recv Codepage=943, Application Char
Codepage=943, Application Graphic Codepage=943 )
[1025232907.500547 - 06/28/2002 11:55:07.500547]

[1025232907.525059              -              06/28/2002              11:55:07.525059]
SQLDriverConnect( szConnStrOut="DSN=SAMPLE;UID=db2inst1;PWD=*****;MODE=SHARE;REPAINT=2;PAT
CH2=6;PATCH1=1024;LOBMAXCOLUMNSIZE=1048575;LONGDATACOMPAT=1;DBALIAS=SAMPLE;",
pcbConnStrOut=132 )
[1025232907.531792 - 06/28/2002 11:55:07.531792]       <--- SQL_SUCCESS    Time elapsed -
+3.513030E+000 seconds
[1025232907.532956 - 06/28/2002 11:55:07.532956] ( DSN="SAMPLE" )
[1025232907.540386 - 06/28/2002 11:55:07.540386]
[1025232907.541166 - 06/28/2002 11:55:07.541166] ( UID="db2inst1" )
[1025232907.543130 - 06/28/2002 11:55:07.543130]
[1025232907.545062 - 06/28/2002 11:55:07.545062] ( PWD="*****" )
[1025232907.546495 - 06/28/2002 11:55:07.546495]
[1025232907.547205 - 06/28/2002 11:55:07.547205] ( MODE="SHARE" )
[1025232907.549634 - 06/28/2002 11:55:07.549634]
[1025232907.550341 - 06/28/2002 11:55:07.550341] ( REPAINT="2" )
[1025232907.551693 - 06/28/2002 11:55:07.551693]
[1025232907.558645 - 06/28/2002 11:55:07.558645] ( PATCH2="6" )
[1025232907.560060 - 06/28/2002 11:55:07.560060]
[1025232907.560766 - 06/28/2002 11:55:07.560766] ( PATCH1="1024" )   ←= 反映されている
[1025232907.563598 - 06/28/2002 11:55:07.563598]
[1025232907.564336 - 06/28/2002 11:55:07.564336] ( LOBMAXCOLUMNSIZE="1048575" )
[1025232907.566631 - 06/28/2002 11:55:07.566631]
[1025232907.567360 - 06/28/2002 11:55:07.567360] ( LONGDATACOMPAT="1" )
[1025232907.568719 - 06/28/2002 11:55:07.568719]
[1025232907.570454 - 06/28/2002 11:55:07.570454] ( DBALIAS="SAMPLE" )
[1025232907.571850 - 06/28/2002 11:55:07.571850]
```

【JDBC Type4ドライバーの場合の設定】

JDBC Type4ドライバーの JDBCトレースは、db2.cli などの設定変更では取得できず、ソースコード (*.java) を修正する必要があります。

JDBC 2 コネクティビティを使用している場合:

Com.ibm.db2.jcc.DB2SimpleDataSource オブジェクトの setJCCLogWrite() メソッドと

setTracelevel() メソッドを使ってトレースを取得します。

JDBC 1 コネクティビティを使用している場合:

java.sql.DriverManager.setLogWrite() メソッドを使用する、あるいは、下記のように、JDBC 1 データベース URL に traceFile, traceLevel キーワードを埋め込むことでトレースを取得することができます。

String databaseURL = “jdbc:db2://host:50000/sample:traceFile=/tmp/jdbc4trc.txt;traceLevel=”+(com.ibm.db2.jcc.DB2BaseDataSource.TRACE_ALL)+”;”;

この指定では、すべてのアクティビティについてトレースを取得します。まず最初には、この指定でトレースを取得してください。

トレースレベルを変更して取得対象を絞るすることが可能です。たとえば、以下の指定では、DRDA のデータフローと接続についてのみトレースを取得します。

String databaseURL = “jdbc:db2://host:50000/sample:traceFile=/tmp/jdbc4trc.txt;traceLevel=”+(com.ibm.db2.jcc.DB2BaseDataSource.TRACE_DRDA_FLOWS |
com.ibm.db2.jcc.DB2BaseDataSource.TRACE_CONNECTS)+”;”;

JDBC 1/2 コネクティビティともに、DB2Connection.setJCCLogWriter(java.io.PrintWrite | logWriter, int traceLevel) メソッドを使ってトレースの on / off、ログライターの変更やアクティビコネクションのトレースレベルの変更が可能です。

標準の javax.sql.Datasource.stLogWriter(java.io.Printer | logWriter)メソッドを使用している場合には、デフォルトのトレースレベル(すべて on)の最も詳細なトレースを取得します。

setJCCLogWriter メソッドで null を設定する、あるいは、logWriter に null をし指定すると、デフォルトではトレースは停止します。

なお、設定できる、トレースレベルのプロパティは以下のとおりです。

int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_NONE = 0x00
int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_CONNECTION_CALLS
int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_STATEMENT_CALLS
int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_RESULT_SET_CALLS
int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_DRIVER_CONFIGURATION
int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_CONNECTS
int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_DRDA_FLOWS
int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_RESULT_SET_META_DATA
int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_PARAMETER_META_DATA
int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_DIAGNOSTICS
int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_SQLJ
int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_ALL = 0xFFFFFFFF

使用例につきましては、以下を参照してください。

http://www.ibm.com/software/data/db2/udb/ad/v8/java/t0010268.htm

2-2-3 DB2 コネクトトレース(ddcstrc)

【説明】

DB2 コネクトトレースは、DB2 コネクト製品の問題を判断するのに使用します。

使用方法は『DB2コネクト使用者の手引き』を参照ください。

なお、このトレースは、DB2トレース (db2trc) と同時に取得できないことに注意してください。

2-2-4 DRDA トレース(db2drdat)

DRDA(Distributed Relational Database Architecture) アプリケーション・リクエスター (AR) および DRDA アプリケーション・サーバー (AS) の間で交換された DRDA データ・ストリームを取り込むことができます。このツールは通常、問題判別のために使用されますが、クライアント / サーバー環境でパフォーマンス調整を行う時にも使用できます。なお、これまで、DRDA は、主に、ホスト DB2 との通信で使用されて来ました。しかし、V8.1 よりクライアント(V8.1)とサーバー（V8.1）の間の通信アーキテクチャーが DRDA に変更になったため、分散環境の通常のクライアント・サーバーの構成において、このトレースを使って分析をすることが可能となりました。なお、このトレースは、クライアント、および、サーバーの両方で取得できます。

使用方法は『コマンド・リファレンス』を参照ください。

なお、このトレースは、DB2トレース (db2trc) と同時に取得できないことに注意してください。

2-3 OS のエラー情報

本節はAIXに特化した内容を記載いたします。

2-3-1 エラーログ

【説明】

OS がハードウエア、ソフトウエアのエラーを報告するために用います。errdaemon というデーモンが記録します。エラーログに記録された情報は errpt コマンドで参照することができます。errpt コマンドの使い方やオプションはAIXのマニュアルを参照ください。

【出力例】

```
$ errpt

IDENTIFIER TIMESTAMP  T C RESOURCE_NAME  DESCRIPTION
C60BB505   0610202302 P S SYSPROC        ソフトウェア・プログラムが異常終了
C60BB505   0610202202 P S SYSPROC        ソフトウェア・プログラムが異常終了
2A9F5252   0610081202 P H tok0           ワイヤ失敗
C60BB505   0607171402 P S SYSPROC        ソフトウェア・プログラムが異常終了
```

2-3-2 syslog

syslogdというデーモンの働きによりシステムの運用状況を管理者に通知したり、運用記録を保存したりすることができます。syslogの動作は /etc/syslog.conf という設定ファイルで制御します。このファイルを設定することで、例えばユーザーがシステムをどのように使用しているか、各デーモンやカーネルなどがどのように動作しているかなどといったシステムの運用状態を把握できるようになります。またこれらの動作状況の記録先や通知先も /etc/syslog.conf ファイルで設定します。DB2 のアラート・ログはこのsyslogに書き出されます。2-1-2 アラート・ログを参照してください。

2-3-3 core

【説明】

システムコアファイルは、DB2に依存したファイルではなく、プログラムが異常終了した場合に出力される UNIX のファイルであり、異常終了した時の関数名などが出力されています。コアファイルの名前は"core"で、アプリケーションが実行されているディレクトリーに出力されます。

プログラムが異常終了すると、システムはコアファイルを作成し、終了したプロセスのメモリー・イメージを保管します。メモリー・アドレス違反、不法な指示、バス・エラー、およびユーザー生成の終了シグナルなどのエラーが原因で、コアファイルのダンプが起きます。

システムコアファイルは、DB2 コアファイル(2-1-3 節参照)とは別であることに注意してください。

【確認手順】

① システムコアファイルの検索とその作成日の確認

```
$ find 〈path〉 -name core -print
$ cd 〈core の出力 path〉
$ ls -l core
```

② システムコアファイルが発生するとエラーログに CORE_DUMP のエラーラベル名で記録されるので、その記録より実行プログラム名は判別可能です。

③ dbx システム・コマンドを使用して、どの関数がコアファイルの作成の原因となったかを判別できます。これは、データベースマネージャーまたは DB2 コネクトがエラーを起こしたかどうかを識別したり、問題の原因はオペレーティング・システムまたはアプリケーションのエラーなのかどうかを識別するのに役立つ簡単な検査です。

コアファイル・ダンプが起きる原因となった関数を判別するためには、次のコマンドを入力してください。

```
$ dbx 〈program_name〉 〈core_filename〉
```

ここで、<*program_name*>は、異常終了したプログラムの名前であり、<*core_filename*>は、 コアファイル・ダンプを含んでいるファイルの名前です。 <*core_filename*>パラメーターはオプションです。 これを指定しなければ、デフォルト名の"core" が使用されます。 dbx は、core ファイルによって該当プログラムのコンパイル時の -g オプションの有無を認識しており、 -g なしでコンパイルされている場合は、dbx コマンドの機能が一部制限されます。dbx コマンドを終了させるためには、dbx プロンプトに quit を入力します。 dbx コマンドには、 この節で説明されているよりもさらに多くの関数を提供します。 それらを知るには、UNIX ベースのコマンド・プロンプトから man dbx を入力してください。

なお、ご使用の環境で dbx が導入されていない場合には、dbx は、bos.adt.debug ファイルセットに含まれおりますので、そのファイルセットを導入してください。

【解析手順】

次の例は、dbx コマンドを使用して、 "main" という名前のプログラムについてのコアファイルを読み取る方法を示しています。

① コマンド・プロンプトから、次のように入力します。

```
$ dbx main
```

② 以下と似た出力が、画面に表示されます。

```
dbx version 3.1 for AIX.
Type 'help' for help.
   reading symbolic information ...
 [using memory image in core]
segmentation.violation in freeSegments at line 136
136 (void) shmdt((void *) pcAdress[i]);
```

上記の例では、メモリー・ダンプを引き起こした関数の名前は "freeSegments" です。 関数名が "db2"、"sql"、または "ddcs" で始まっている場合、 エラーがデータベースマネージャーまたは DB2 コネクト製品で起きた可能性があることを示します。

③ 障害個所へのプログラム・パスを表示するために、dbx プロンプトから where を入力します。

```
(dbx) where
```

```
freeSegments(numSegs = 2, iSetId = 0x2ff7f730, pcAddress = 0x2ff7f758, line136
in "main.c"
main (0x1, 2ff7f7d4), line 96 in "main.c"
```

この例では、"main.c" の第 96 行から呼び出された freeSegments の第 136 行でエラーが起きました。

④ dbx コマンドを終了させるためには、dbx プロンプトに quit を入力します。

```
(dbx) quit
```

3 問題判別時に役立つツールのご紹介

コマンドの詳細は『コマンド解説書』を参照ください。

3-1 DB2 の稼動状況を確認するための主なツール

3-1-1 LIST APPLICATIONS (DB2 CLP コマンド)

【説明】

活動中のすべてのデータベース・アプリケーション情報を表示します。アプリケーション・プログラム名、許可 ID (ユーザー名)、アプリケーション・ハンドル、アプリケーション ID、およびデータベース名を標準出力に出力します。なお、show detail オプションをつけることにより、現在のアプリケーションの状況、db2 エージェントのプロセスIDなどの追加情報を取得できます。

【使用例】

```
$ db2  list  applications
許可 ID  アプリケーション  アプリケーション・   アプリケーション ID          DB      エージェントの
         名               ハンドル                                        名      数
-------- -------------- ---------- ----------------------------- -------- -----
DB2INST1 db2bp          7          *LOCAL.db2inst1.080D92114809   SAMPLE   1
```

3-1-2 LIST TABLESPACES (DB2 CLP コマンド)

現行データベースの表スペースのリストを表示します。DMS 表スペースであれば、SHOW DETAIL オプションを追加することにより、表スペースの使用中のサイズも表示されます。

3-1-3 LIST TABLESPACE CONTAINERS (DB2 CLP コマンド)

指定した表スペースのコンテナーのリストを表示します。

3-1-4 db2tbst (DB2 システム・コマンド)

【説明】

前述の LIST TABLESPACES コマンドの出力結果の状況フィールドに表示される 16 進数の値をテキスト記述したものを提供します。

【使用例】

```
$ db2tbst 0x0000
State = Normal
```

3-1-5 ストレージ管理ツール

ストレージ管理ツールは、コントロール・センターを介して使用することができます。このツールからは、特定のデータベース、データベース・パーティション・グループ、または表スペースに関するストレージのスナップショットを表示するストレージ管理ビューにアクセスすることができます。

以下のオブジェクト別に情報を取り込むことができます。

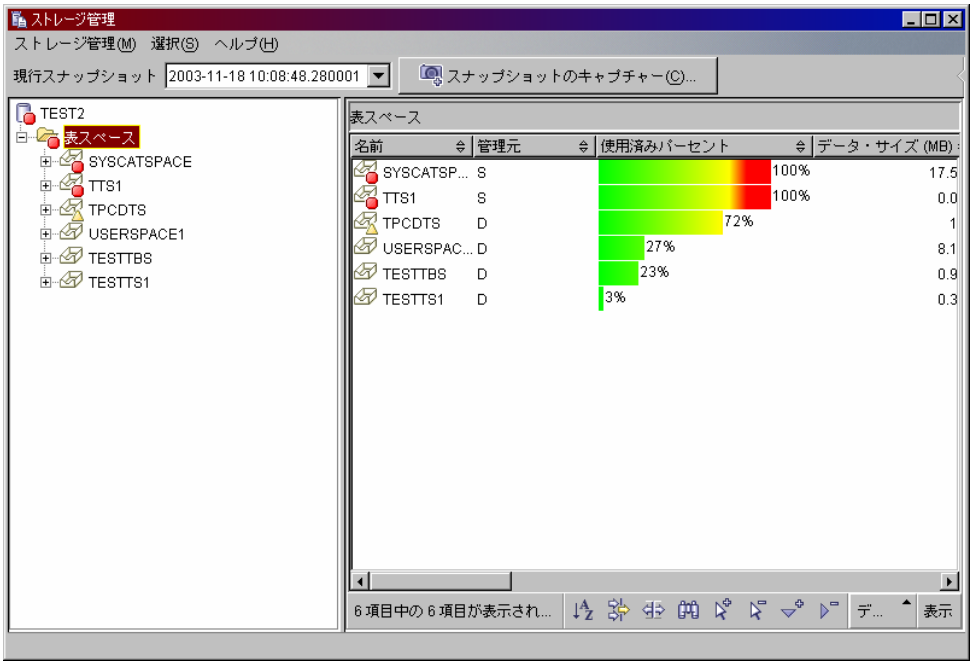
- 表スペースの場合、指定表スペースの有効範囲のもとで定義されている表、索引、およびコン

テナーに関する情報が、システム・カタログとデータベース・モニターを情報源として表示されます。

- データベースまたはデータベース・パーティション・グループの場合、特定のデータベースまたはデータベース・パーティション・グループに定義されているすべての表スペースに関する情報が表示されます。
- データベースの場合、データベース内のすべてのデータベース・パーティション・グループに関する情報も収集されます。

また、ストレージ管理ビューからは、データ・スキュー、スペース使用量、および索引クラスター率のしきい値を設定することもできます。ターゲット・オブジェクトが指定しきい値を超えた場合、警告またはアラーム・フラグで通知されます。

※出力例



3-1-6 スナップショット・モニター

【説明】

ある一時点でのデータベース活動状況を収集します。

モニター単位、また、モニター記録スイッチの設定により表示できる情報が異なります。

ーモニター単位

Database Manager ：DB2 インスタンスへのデータベース接続および Agent の使用状況などを得る。

Database ：データベースに発生しているロックの情報、バッファープールの情報、全アプリケーション稼働の総合情報などを得る。データベースの総合的なパフォーマンス解析に役立つ。

Application ：アプリケーションごとの稼働状況、読み取り行数、更新行数などの活動情報を得る。DB2 アプリケーションが稼働中であるかの確認に役立つ。

Tablespace：表スペースごとに入出力活動とバッファープール活動などを得る。

Table ：アクセスされている各種の表を表示。

Lock ：ロック待機、ロックを保持しているアプリケーションの特定などに用いる。

Bufferpool ：バッファープールごとにバッファーヒット率を計算できるため、パフォーマンス解析等に役立つ。

ーモニター記録スイッチ

スナップショットモニターでは基本情報として取得できる情報と、モニタースイッチ記録スイッチをONにすることにより収集可能な情報があります。取得したい情報が含まれる最小限のモニタースイッチを ON にしてください。スイッチをONにする際には必ずモニタリングを行う接続で行います。 スイッチをONにしても、別の接続で行われた処理に関する情報は取得することができません。デフォルトはデータベース構成パラメーターDFT_MON_xx の値です。

BUFFERPOOL	バッファープール活動情報
LOCK	ロック情報
SORT	ソート情報
STATEMENT	SQL ステートメント情報
TABLE	表活動情報

どのような情報が、どのモニター単位で、どのモニター記録スイッチで得られるのかは、『システム・モニター 手引きおよび解説書』を参照ください。

【使用例】

① モニター記録スイッチの現在の状況表示。

```
$ db2 get monitor switcies
                モニター記録スイッチ
DB（データベース）パーティション番号のスイッチ・リスト       0
バッファー・プール活動情報                (BUFFERPOOL) = OFF
ロック情報                                  (LOCK)      = OFF
ソート情報                                  (SORT)      = OFF
SQL ステートメント情報                      (STATEMENT) = OFF
表活動情報                                  (TABLE)     = OFF
タイム・スタンプ情報を取る                  (TIMESTAMP) = ON  06/14/2004 02:46:12.285085
作業単位情報                                (UOW) = OFF
```

② 取得したい情報を得るためのモニター記録スイッチを ON に設定。

```
$ db2 update monitor switches using <switch-name> on
    <switch-name> に以下のスイッチ名が使用できます。
 BUFFERPOOL              バッファープール活動情報
 LOCK                    ロック情報
 SORT                    ソート情報
 STATEMENT               SQL ステートメント情報
 TABLE          表活動情報
 TIMESTAMP               タイム・スタンプ
```

UOW　　作業単位情報

③ カウンター・タイプ(値が積算されていく項目)の項目値をリセット。

$ db2 reset monitor for database 〈*database-alias*〉

④ スナップショットの取得。

<良く使うスナップショット取得コマンドの例>

db2 get snapshot for all on〈*database-alias*〉

:該当データベース全てに関する一般統計を提供

db2 get snapshot for locks on 〈*database-alias*〉

:該当のデータベースのロック状況スナップショットのみを表示。ロック待ち状況の判断に有効。

db2 get snapshot for database manager

:データベース・マネージャー(インスタンス)の統計情報を提供。

db2 get snapshot for all applications

:インスタンス内のデータベースに接続された活動アプリケーション全てに関する情報を提供。

【出力例】 基本のモニター記録スイッチでの、データベース・マネージャー情報の出力結果

```
$ db2 get snapshot for database manager
                データベース・マネージャー・スナップショット

 ノード・タイプ                                  = ローカルとリモート
 ・クライアントを持つデータベース・サーバー
 インスタンス名                                  = db2inst1
 DB2 インスタンスのノード数                      = 1
 データベース・マネージャー状況                  = Active

 製品名                                          =
 製品 ID                                         =
 サービス・レベル                                =

 ソート・ヒープ割り振り                          = 0
 ポストしきい値ソート                            = 収集されませんでした
 パイプ済みソート要求                            = 0
 パイプ済みソート受け入れ                        = 0

 データベース・マネージャー開始タイム・スタンプ  = 06/26/2002 17:38:32.473367
 最後のリセット・タイム・スタンプ                =
 スナップショット・タイム・スタンプ              = 06/28/2002 12:38:30.268640
 DB マネージャーへのリモート接続                 = 0
 DB マネージャーで実行中のリモート接続           = 0
 ローカル接続                                    = 1
 DB マネージャーで実行中のローカル接続           = 0
 アクティブ・ローカル・データベース              = 1

 登録済みエージェント用最高水準点                = 2
 トークン待機中のエージェント用最高水準点        = 0
 登録済みエージェント                            = 2
 トークン待機中のエージェント                    = 0
 アイドル・エージェント                          = 0

 コミット済み専用メモリー (バイト)               = 245760
```

```
ノード 0 のスイッチ・リスト
バッファー・プール活動情報          (BUFFERPOOL) = OFF
ロック情報                          (LOCK)      = OFF
ソート情報                          (SORT)      = OFF
SQL ステートメント情報              (STATEMENT) = OFF
表活動情報                          (TABLE)     = OFF
作業単位情報                              (UOW) = OFF

プールから割り当てられたエージェント                    = 3
空のプールから作成されたエージェント                    = 2
別のアプリケーションからスチールされたエージェント      = 0
調整エージェント用最高水準点                            = 2
最大エージェント・オーバーフロー                        = 0
ヒープしきい値超過後のハッシュ結合                      = 0

ゲートウェイ接続合計数                                  = 0
ゲートウェイ接続現在数                                  = 0
ホスト応答待機中のゲートウェイ接続                      = 0
クライアント要求待機中のゲートウェイ接続                = 0
ゲートウェイ非アクティブ接続プール・エージェント        = 0
ゲートウェイ接続プール・エージェント・スチール          = 0
```

3-1-7 イベント・モニター

【説明】

イベント・モニターの開始時点から停止までの各種イベント別の情報を収集します。どのイベント・タイプの情報を収集したいかにより、イベント・モニターの設定が異なります。

<イベント・タイプ>

Deadlock(デッドロック発生時のアプリケーションとロック情報)

Statements(SQL レベルでの開始・終了時刻や CPU 時間など)

Transactions(UOW レベルでの開始・終了時刻や CPU 時間など)

Connections(アプリケーションレベルでのカウンター情報)

Database(データベースレベルでのカウンター情報)

Bufferpools(バッファープールレベルでのバッファープール、プリフェッチャーなどの情報)

Tablespace(表スペースレベルでのバッファープール、プリフェッチャーなどの情報)

Tables(表レベルでの読み込み／書き込み)

収集するイベント・タイプにより、パフォーマンスの劣化は起こり得ますので注意が必要です。ただ、トレースよりは負荷は軽いので、再現性の低い(再現方法が不明な)障害解析にむいています。また、イベント・ファイルのファイルシステム容量が足りなくならないようご注意ください。

DB2 V8 からは、「デッドロックの詳細」イベント・モニターが、新規に作成されるデータベースごとにデフォルトで作成されます。 このイベント・モニターは DB2DETAILDEADLOCK という名前で、データベースが活動化されると開始され、データベース・ディレクトリー内のファイルに書き込みます。このイベント・モニターによるオーバーヘッドは、これをドロップすることによって回避することができます。

【使用例】 全イベント・タイプを取得する例

以下はあくまで例となります。全イベント・タイプを取得した場合、パフォーマンスに影響するためご注意ください。

＜イベント・モニターの作成＞

① 対象データベースに接続。

$ db2 connect to <*database-alias*>

② イベント・モニターの結果出力先ディレクトリー作成。

この例では「/work/monitor」を出力先とし、インスタンスオーナーで作業すると仮定します。

$ mkdir /work/monitor

$ chown -R $DB2INSTANCE /work/monitor

※取得データ量は、アプリケーションの数、取得時間、などに大きく依存しますが、一般的に50Mb程度の空き容量を確保する必要があります。

③ イベント・モニターの作成。

$ db2 "create event monitor <*monitor-name*> for database,tables,deadlocks,tablespaces,bufferpools,connections,statements,transactions write to file '/work/monitor' replace"

※<*monitor-name*>: 任意のイベントモニター名

④ イベント・モニターの開始。

$ db2 set event monitor <*monitor-name*> state = 1

※ ③だけではイベントの取得は開始されません。

⑤ 障害の再現。

⑥ イベント・モニターの停止。

$ db2 set event monitor <*monitor-name*> state = 0

⑦ 取得イベント・データのフォーマット。

$ db2 flush event monitor <monitor-name>

$ db2evmon -path /work/monitor ＞ **出力ファイル**

※イベント・モニターの結果が一部キャッシュ上に残っている可能性があるので、flush event monitor SQL ステートメントにて、ディスクに書き出します。

※出力ファイルは容量が大きくなりますので、十分に空き容量のあるパスを使用して下さい。

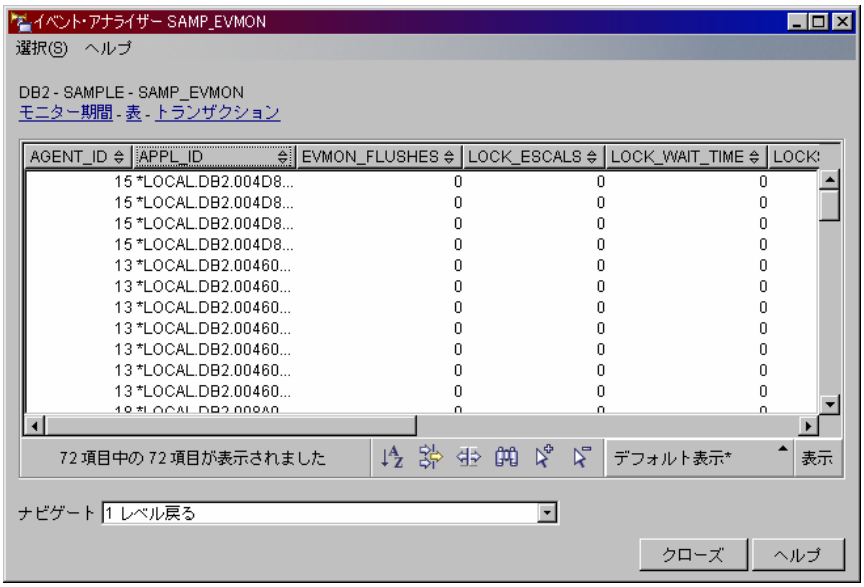
⑧ イベント・モニターの削除。

$ db2 drop event monitor <*monitor-name*>

障害発生頻度が低い場合、イベント・モニターの出力ファイルが増大します。そのような場合は、定期的に④(開始)と⑥(停止)を繰り返すとよいでしょう。

DB2 V8 から、イベント・モニターは、ファイルやパイプの代わりに SQL 表にデータを書き込むことができるようになっています。イベント・アナライザーを使用することにより、 DB2 イベント・モニターが生成し表に送ったパフォーマンス・データをトレースできます。

※イベント・アナライザーによる出力例



3-1-8 ヘルス・モニター、ヘルス・センター

【説明】

DB2 V8 には、DB2 システムの情報をモニターするための新しいフィーチャーであるヘルス・モニターとヘルス・センターが導入されています。

ヘルス・モニターは、ユーザー対話がなくてもインスタンスの状態を常にモニターするサーバー・サイドのツールです。ヘルス・モニターは、定義されているしきい値を超えていること (たとえば、使用可能なログ・スペースが十分でないなど) を発見したり、オブジェクトについての異常な状態 (たとえば、インスタンスが故障しているなど) を検出したりする場合に、アラートを出します。

ヘルス・モニターは、パフォーマンスを特定の側面から評価するためにヘルス・インディケーターを使用します。ヘルス・インディケーターは、データベースの正常な稼動を評価するための個々のしきい値です。

ヘルス・インディケーターのカテゴリーには以下のものがあります。

表スペース・ストレージ

ソート

データベース管理システム

データベース

ロギング

アプリケーション並行性

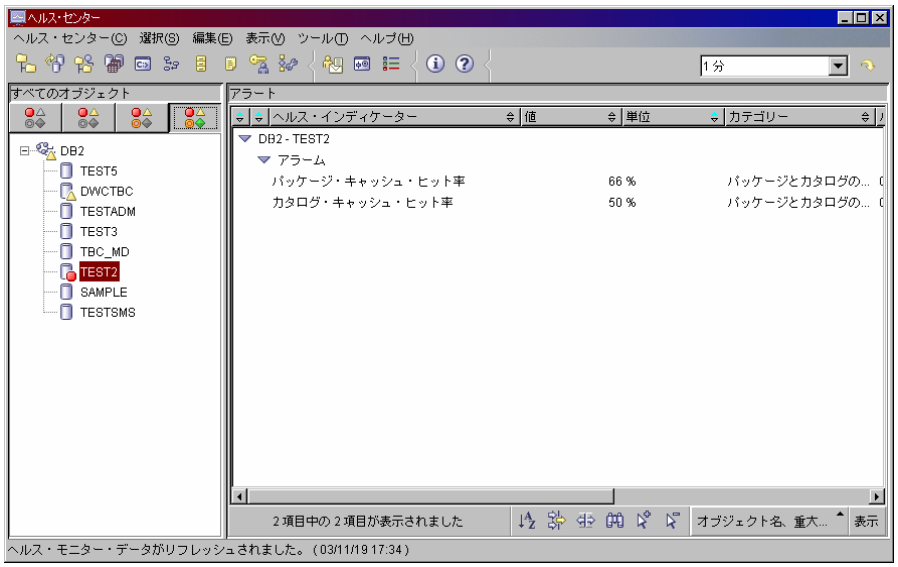
パッケージおよびカタログ・キャッシュ、およびワークスペース

メモリー

ヘルス・センターは、ヘルス・モニターへのグラフィカル・インターフェースを提供します。これは、ヘルス・モニターを構成するためと、ロールアップされたインスタンスおよびデータベース・オブジェクトのアラート状態を表示するために使用します。ヘルス・モニターのドリルダウン機能を使用すれば、現行アラートの詳細にアクセスして、アラートの解決方法が記述された推奨処置のリストを表示することができ

ます。

ヘルス・センターでの出力例



3-1-9 Explain 機能

ある照会が DB2 によってどのように実行されるかを知りたい場合は、アクセスパスを解析する必要があります。Explain 機能は DB2 が SQL 文を実行するためにデータにアクセスする方法についての情報を提供します。

SQL 処理にボトルネックがあると思われる場合には良く使う機能です。

大きく分けると、Explain 表に Explain 情報を収集しそれを分析する方法と、Explain 表を使わない方法があります。

3-1-9-1 Explain 表への情報収集

Explain 表はあらかじめ用意しておいてください。 $INSTHOME/sqllib/misc の下にあるEXPLAIN.DDLを情報収集するユーザーで実行することにより用意できます。Explain 情報の中で、後に Visual Explain で分析するための情報を Explain スナップショット情報といいます。

Explain情報をExplain表に収集するには、下記[a] ～[c]の3つの方法があります。情報を収集後の分析方法については3-1-7-2 Explain表に収集した情報を元にアクセスパスを解析 を参照ください。

[a] EXPLAIN 文

【説明】

単一の動的 SQL 文の Explain 情報を収集したい場合に使えます。問題のある SQL 文が特定できている場合などに使用します。

【使用例】 Explain 情報と Explain スナップショット情報を取得する

実行する SQL 文の前に、EXPLAIN 文を追加します。

```
$ db2 EXPLAIN ALL WITH SNAPSHOT FOR <sql-statement>
```

[b] CURRENT EXPLAIN MODE/CURRENT EXPLAIN SNAPSHOT 特殊レジスター

【説明】

単一あるいは複数の動的 SQL の Explain 情報を収集したい場合に使えます。単体テスト時などに、プログラム中の全動的 SQL の Explain 情報収集する場合などには便利です。

【使用例】 SQL は実際には実行しないが Explain 情報と Explain スナップショット情報を取得します。

CURRENT EXPLAIN MODE と CURRENT EXPLAIN SNAPSHOT 特殊レジスターを設定します。

```
$ db2 SET CURRENT EXPLAIN MODE EXPLAIN
$ db2 SET CURRENT EXPLAIN SNAPSHOT ＥＸＰＬＡＩＮ
$ db2 “<sql-statement>”
$ db2 “<sql-statement>”
```

[c] PREP（BIND）の EXPLAIN/EXPLSNAP オプション

【説明】

PREP(BIND)の EXPLAIN/EXPLSNAP オプションを用いて、PREP(BIND)時に静的 SQL 文の Explain 情報を取得、あるいは、実行時に動的 SQL 文の Explain 情報を取得することができます。

単体、統合、システムテストのそれぞれの局面で、全パッケージの静的 SQL の Explain 情報を収集して保存しておきたい場合などに便利です。

【使用例】 Explain 情報と Explain スナップショット情報を取得する

EXPLAIN/EXPLSNAP オプションを設定してプリコンパイルします。

```
$ db2 PREP <pgm-source> EXPLAIN ALL EXPLSNAP ＡＬＬ
```

3-1-9-2 Explain 表に収集した情報を元にアクセスパスを解析

Explain表をそのまま照会することも可能ですが、テキスト形式、及びGUI形式に情報をフォーマットするためのツールがあります。いずれもあらかじめExplain表に情報収集しておいてください。（3-1-7-1 節 Explain表への情報収集参照）

[a] db2exfmt (DB2 システム・コマンド)

【説明】

Explain 情報をテキスト形式でフォーマットするものです。もっとも詳細な情報を取得することができます。アクセスプランを分析する際には、このツールを使用することをお勧めします。

db2exfmt [[-1] [-d <database-alias>] [-e <schema>] [-f O] [-h] [-l] [-n <name>] [-o <outputfile>] [-s <schema>] [-t]] [-u <user> <pw>] [-w <timestamp>] [-# <sectnbr>]

-w <timestamp> : Explain タイムスタンプ。最新の Explain 要求を取得するのは-1 を指定。

-l : パッケージ名を小文字又は大小混合文字のいずれかで指定することができます。この-l オプションを指定しないと、パッケージ名は大文字に変換されます。

-# <sectnbr> : ソース内のセクション番号。全てのセクションを指定するには、ゼロを使用します。

【使用例】

① Explain表に情報収集 (CURRENT EXPLAIN MODE特殊レジスターを用いた例)

```
$ db2 set current explain mode explain
$ db2 <sql-statement>
```

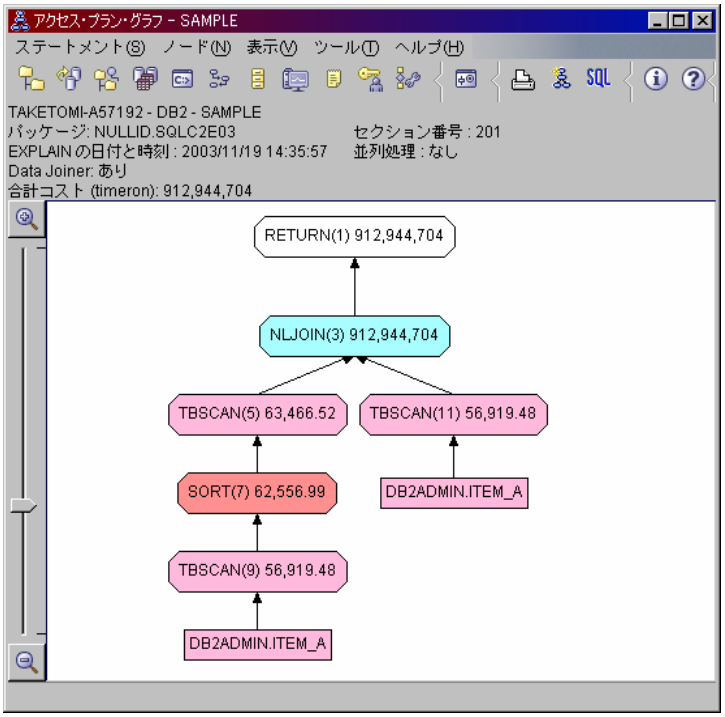
② db2exfmtコマンドで①で取得したExplain情報をテキスト形式でファイルへ出力。

```
$ db2exfmt -d <database-alias> -g TIC -w -1 -n % -s % -# 0 -o <outputfile>
```

[b] Visual Explain

Explain スナップショット情報を GUI 形式でフォーマットするものです。Visual Explain は一般にコントロール・センターから呼び出しますが、db2vexp コマンドも使用できます。Explain 表が存在しないときに Visual Explain を実行すると、Explain 表は自動作成されます。

出力例



3-1-9-3 Explain 表を使用しないでアクセスパスを解析

Explain 表に情報を収集することなくアクセスパスを解析するため Explain 機能として 2 つ提供しています。

[a] db2expln (DB2 システム・コマンド)

【説明】

静的 SQL 文のアクセスパス情報は、パッケージの一部として PREP(BIND)時に生成され、システム・カタログに保管されています。この情報を直接見るには、db2expln コマンドを使用します。生成された実際のアクセスパスを参照することにより、実行時にどのような操作が行われるかに関する比較的コンパクトなリポートを提供します。本番稼動中の静的 SQL のアクセスパスを確認するのに役立ちます。

dynexpln ツールは、後方互換性のために今でも使用可能です。しかし、db2explnのdynamic-optionsを使用して、 dynexpln のすべての機能を実行できます。db2expln の dynamic-options を使用するとき、ステートメントは真の動的 SQLとして準備され、生成されるプランは SQL キャッシュからExplain されます。この explain-output メソッドは、ステートメントを静的 SQL として準備するdynexpln よりも正確なアクセス・プランを提供します。これにより、パラメーター・マーカーなど動的 SQL だけで使用できる機能も使用可能になります。

【使用例】 全パッケージの情報を出力する例

$ db2expln -d <*database-alias*> -p % -c % -s 0 -o <*outputfile*>

```
Access Table-name = SYSIBM.SYSCOLUMNS  ID = 3   <- アクセスされる Regular table と SYSCAT.TABLES 中の TABLEID
|  #Columns = 15                               <- 使われる列数
                                               <- １行より多くアクセスされる１行のときはここに"Single Record"と表
示される
                                               <- 表にアクセスする isolation level は package のものと同じ
                                                  違う時は"Isolation Level: XXXX"と出る
                                                <- Scan の方向は forwardreverse のときは"Scan Direction = Reverse"
と出る
|  Index Scan:  Name = SYSIBM.IBM01  ID = 1     <- インデックスを使って Row はアクセスされる
                                                  アクセスされるインデックス名と SYSCAT.INDEXES の ID
|  |  #Key Columns = 2                                  <- インデックスの Key のうちの２列が使われる
                                               <- 表データはアクセスされる
                                                  インデックスのみのアクセスの時"Index-Only Access"と出る
|  |  Data Prefetch: None                              <- Data Prefetch は使われない
|  |  Index Prefetch: None                             <- Index Prefetch は使われない

|  Lock Intents
|  |  Table: Intent None                               <- Table Level Lock Type
|  |  Row  : None                               <- Row Level Lock Type
|  Insert Into Sorted Temp Table  ID = t1               <- temporary table は sort される
                                                  sort されない時は"Create/Insert Into Temp table ID = tn"が出る
|  |  #Columns = 13                             <- temporary table に insert される列数
|  |  #Sort Key Columns = 1                            <- sort で使われるキーの列数
|  |  |  Key 1: COLNO (Ascending)               <- sort で使われるキーの列名
|  |  Sortheap Allocation Parameters:
|  |  |  #Rows     = 3                          <- sort する row 数
|  |  |  Row Width = 48                                <- sort する row の Width(byte)
|  |  Piped                                            <- Pipe sort
Sorted Temp Table Completion  ID = t1
Access Temp Table  ID = t1                             <-アクセスされる temporary table と db2expln がアサインし
た ID
|  #Columns = 13                             <-使われる列数
|  Relation Scan                             <- table を直接読む relation scan
|  |  Prefetch: Eligible                             <- Prefetch は使われる(Applicable)
Return Data to Application
|  #Columns = 15                             <-使われる列数
|  |  Sortheap Allocation Parameters:
|  |  |  #Rows     = 3                          <- sort する row 数
|  |  |  Row Width = 48                         <- sort する row の Width(byte)
|  |  Piped                                    <- Pipe sort
Sorted Temp Table Completion  ID = t1
Access Temp Table  ID = t1                             <-アクセスされる temporary table と db2expln がアサインし
た ID
|  #Columns = 13                             <-使われる列数
|  Relation Scan                             <- table を直接読む relation scan
```

[b] dynexpln (DB2 システム・コマンド)

【説明】

dynexpln コマンドはパラメーター・マーカーを用いていない動的 SQL 文に対して簡易的な Explain 機能を実行します。パラメーター・マーカーを使用している場合は db2exfmt の使用をお奨めします。内部的には、入力の動的 SQL 文を擬似静的 SQL 文に変換して疑似パッケージを作成し、それを db2expln を用いて情報を出力しています。そのため、情報は必ずしも正確ではないこともありますが、簡易的に特定の SQL のアクセスパスを確認するには便利です。V8 より、この機能は、db2expln に入りましたので、今後は、db2expln の方を使用してください。

【使用例】

① 対象の SQL 文をファイルに保存しておきます。

```
select * from table1 ;
```

② ①のファイルを dynexpln を用いて分析します。

```
$ dynexpln -d <database-alias> -f <file-name> -o <outputfile> -z  ';'
```

出力結果は前述の db2expln の出力結果と類似しています。

3-1-10 db2bfd (DB2 システム・コマンド)

【説明】

バインド・ファイルを作成する際に使用したバインド・ファイル・ヘダー(プリコンパイル・オプションが判別可能)を表示するだけではなく、バインド・ファイル内の SQL ステートメント、ホスト変数を調べるためにも使用できます。

【使用例】バインド・ファイル・ヘッダーを表示する例　(db2sampl.bnd が調べたいバインド・ファイル名)

```
> db2bfd -b db2sampl.bnd

db2sampl.bnd:  Header Contents
Header Fields:
Field        Value
-----        -----
releaseNum   0x800
Endian       0x4c
numHvars     7
maxSect              9
numStmt              41
optInternalCnt       4
optCount     10

Name                 Value
-------------------  -----
Isolation Level      Cursor Stability
Creator              "NULLID  "
App Name             "DB2SAMPL"
Timestamp            "CBnQAcKT:2003/10/28 00:16:39:64"
Cnulreqd             Yes
Sql Error            No package
Block                Block All
Validate             Bind
Date                 Default/local
Time                 Default/local

*** All other options are using default settings as specified by the server ***
```

3-1-11 db2flsn (DB2 システム・コマンド)

【説明】

指定したログ順序番号(LSN)で識別されるログレコードを含んでいるファイル名を出力します。LSN はストリング付きの内部 16 進数値を表す 12 バイトのストリングで入力します。

【使用例】

```
$ db2flsn 000000BE0030
```

```
Warning: the result is based on the last known log file size.
The last known log file size is 23 4K pages starting from log extent 2.
Given LSN is contained in log file S0000001.LOG
```

3-1-12 db2support(v7.1　FP4 以降)　(DB2 システム・コマンド)

【説明】

このコマンドは、システム環境を含むDB2サーバーに関する情報を生成します。このプログラムの出力結果は、指定されたパス上の db2support.zip 及び db2support.log という 2 つのファイルとして格納されます。

db2support コマンドは問題発生後、サポートセンターに送付する資料の取得に使用します。

詳細のヘルプは db2support –h により出力される内容を参照してください。

DB に接続し、カレントディレクトリーに情報を収集する場合以下のようなコマンドを実行します。

$ db2support . -d <*database-alias*> -c

【使用例】

2 章 8-1 節共通で必要な情報を参照してください。

3-1-13 メモリー・ビジュアライザー

【説明】

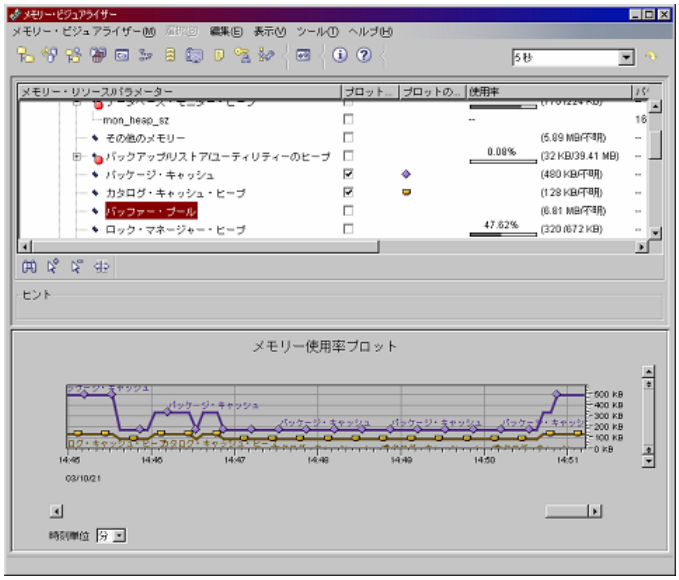
メモリー・ビジュアライザーは、インスタンスおよび階層ツリー内で組織化されたすべてのデータベースの、メモリー関連のパフォーマンスを、データベース管理者がモニターするのに役に立つ、DB2 のGUIツールです。データベース管理者は、メモリー・ビジュアライザー・ウィンドウに、コンポーネントおよび現行のメモリー使用に対して割り振られたメモリーの量の値を表示します。

【使用例】

Windows でメモリー・ビジュアライザーをオープンするには、「スタート」 ->「プログラム」 ->「IBM DB2」 ->「モニター・ツール」 ->「メモリー・ビジュアライザー」 を選択してください。「メモリー・ビジュアライザー・インスタンス選択」ウィンドウがオープンします。

また、コントロール・センターのインスタンスや、ヘルス・センターからも、メモリー・ビジュアライザーは起動できます。

<出力例>



3-1-14　db2mtrk　(DB2 システム・コマンド)

【説明】

データベースやエージェントなどの、メモリー状況レポートを提供します。-i オプションでインスタンス・レベル、データベース・レベルのメモリーを表示します。-p オプションでエージェント ID によってオーダーされた専用メモリー使用量に関する詳細情報を表示します。

【使用例】

```
> db2mtrk -p -i
  トラッキング・メモリー: 2003/12/19 / 18:26:17
  インスタンスのメモリー
     utilh     pckcacheh catcacheh bph       bph       bph       bph
     16.0K     160.0K    48.0K     1.1M      1.1M      640.0K    384.0K
     bph       bph       lockh     dbh       monh      other
     256.0K    192.0K    80.0K     3.1M      176.0K    7.1M
  エージェントのメモリー 2236
     other     apph      appctlh
     16.0K     64.0K     16.0K
```

3-1-15 PCT ツール

【説明】

クライアントとサーバーの間の通信のセットアップをデバッグする際には、DB2 に用意されている Protocol Communications Tester というツールを使用できます。このツールは、sqllib/bin ディレクトリーにある pct で始まる名前のコマンドを使って起動します。このツールは、特定の LAN プロトコルが動作するかどうかや、その LAN プロトコルを使って 2 つのエンドポイントが互いに通信できるかどうかを検証できます。

このツールの使い方については、実行可能ファイルと同じディレクトリーにある readme.pct ファイルを参照してください。PCT ツールでテストできるプロトコルは以下の 3 つです。

NetBIOS

TCP/IP

IPX/SPX

【使用例】

ここでは、TCP/IP バージョンの PCT を使用するための手順を紹介します。IPX/SPX プロトコルや NETBIOS プロトコルで作成した接続をテストする場合の手順もほとんど同じです。主な違いは、ここで「pctt」と入力する箇所に「pctn」や「pcti」を入力するということです。

<サーバー側>

① 次のコマンドを使って、サーバーのホスト名と IP アドレスを確認します。

```
> pctt -r
```

② pctt /gt を実行して、pct.ini ファイルを生成します。

このコマンドは、書き込みアクセス権限があるディレクトリーで実行してください。

③ pct.iniファイルを編集します。たとえば、pct.iniの ServiceName 値または ServerPort 値を、db2 get database manager configurationコマンドの出力の SVCENAMEフィールドと同じ値に設定する必要があります。なお、ServerHostName と ServerIPAddress については、いずれか一方の値を入力するだけでかまいません。ServiceName と ServerPort についても同様で、いずれか一方だけで十分です。

```
[TCPIP]
ServerHostName = ixa006
ServerHostAddress =
ServerIPAddress =
ServiceName =
ServerPort = 50000
```

④ PCT リスナーを開始します。

```
> pctt s
```

<クライアント側>

① pctt /gt を実行して、pct.ini ファイルを生成します。

② pct.ini ファイルを編集します。たとえば、クライアントで CATALOG TCPIP NODE コマンドを実行したときに、サーバーの IPアドレスではなくホスト名を使用した場合は、pct.iniファイルでも同じようにホスト名を使用します。同様に、ポート番号ではなくサービス名を使用した場合は、pct.ini でもサービス名を使用します。

③ pctt c を実行して、pct クライアントを開始します。

④ 成功した場合の結果は、次のようになります。

```
[- Reading configuration parameters
==> pct.ini file was found... using file specified protocol values!
 Protocol Tester values...
   Client/Server          : Client
   Connections            : 1
   Buffer Size            : 500
```

```
   Transaction Iterations : 1
   Trace                  : OFF
   Service                : Send/Recv/Verify
   Keep Connections       : NO
   Delayed Send (secs)    : 0
 Local TCPIP specific values...
          Hostname        : A57177
          HostAddress     : A57177.cswd.co.jp
          IPAddress       : 172.16.231.207

 Server TCPIP specific Values...
   Server Hostname        : ixa006
   Server HostAddress     : ixa006
   Server IPAddress       : 172.16.231.6
   Service Name           :
   Server Port            : 50000

- Initializing the Protocol             Date: 12/08/03  Time: 10:14:39:916
- Connecting to Remote System           Date: 12/08/03  Time: 10:14:39:916
| retcode = <    0>  ----[ TCPIP.socket     ]-----[  SUCCESS  ]------------
| retcode = <    0>  ----[ TCPIP.connect    ]-----[  SUCCESS  ]------------
- Connection established!     1          Date: 12/08/03  Time: 10:14:39:966
- Sending Service Request               Date: 12/08/03  Time: 10:14:39:966
| retcode = <    0>  ----[ TCPIP.send       ]---[ Bytes=  316 ]------------
- Receiving                             Date: 12/08/03  Time: 10:14:39:966
| retcode = <    0>  ----[ TCPIP.receive    ]---[ Bytes=    0 ]------------
- Send/Receive/Verify Data Loop         Date: 12/08/03  Time: 10:14:40: 56
- Sending Data                          Date: 12/08/03  Time: 10:14:40: 56
| retcode = <    0>  ----[ TCPIP.send       ]---[ Bytes=  500 ]------------
- Receiving                             Date: 12/08/03  Time: 10:14:40: 56
| retcode = <    0>  ----[ TCPIP.receive    ]---[ Bytes=    0 ]------------
- Terminating the Connection            Date: 12/08/03  Time: 10:14:40: 56
| retcode = <    0>  ----[ TCPIP.sock_close ]-----[  SUCCESS  ]------------
+-----------------------------------------------+
|  Protocol Connection Test Completed - SUCCESS  |
+-----------------------------------------------+
```

3-1-16 db2gcf　(DB2 システム・コマンド)

【説明】

db2インスタンスを開始、停止、モニターするためのコマンド。一部のHA Cluster Softwareにおいて、そのHA Script内で使用されています。

現在、db2gcfコマンドは、コマンド・リファレンスには記載されていません。詳しくは、以下を参照ください。

http://www-1.ibm.com/support/docview.wss?rs=71&context=SSEPGG&q1=db2gcf&uid=swg21114104&loc=en_US&cs=utf-8&lang=en+en

http://www-3.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/aparlib.d2w/display_apar_details?aparno=IY41505

なお、このコマンドをHA Script内で使用する場合には、V8.1 fixpak6 を使用してください。

【資料例】

```
$ db2gcf -s

Instance  : db2inst1
DB2 State : Available
```

3-2 DB2 の設定状況を確認するツール

3-2-1 GET DATABASE MANAGER CONFIGURATION　(DB2 CLP コマンド)

【説明】

データベース・マネージャー構成パラメーターを取得します。稼動環境を把握しておくためにも、日頃より管理しておくべき情報です。

【使用例】

```
$ db2 get dbm cfg
```

3-2-2 GET DATABASE CONFIGURATION　(DB2 CLP コマンド)

【説明】

データベース構成パラメーター情報を取得します。データベースごとの情報ですので、取得時にデータベース名を指定します。稼動環境を把握しておくためにも、日頃より管理しておくべき情報です。

【使用例】

```
$ db2 get db cfg for <database-alias>
```

3-2-3 db2set　(DB2 システム・コマンド)

【説明】

DB2 レジストリー変数を表示、設定、または除去します。

稼動環境を把握しておくためにも、日頃より管理しておくべき情報です。

【使用例】

<ローカル環境変数を全て表示する例>

```
$ db2set -all
```

- [i] DB2_PINNED_BP=YES
- [i] DB2_SQLROUTINE_COMPILE_COMMAND=xlc -I/home/db2inst1/sqllib/include SQLROUTINE_FILENAME.c -bE:SQLROUTINE_FILENAME.exp -e SQLROUTINE_ENTRY -o SQLROUTINE_FILENAME -L/home/db2inst1/sqllib/lib -ldb2 -q64
- [i] DB2COMM=tcpip
- [i] DB2CODEPAGE=943
- [i] DB2AUTOSTART=YES
- [g] DB2SYSTEM=dmaix2
- [g] DB2ADMINSERVER=dasusr2

<ローカル環境変数を設定する例>

```
$ db2set  DB2COMM=TCPIP
```

<ローカル環境変数の設定を解除する例>

変数を削除するには、値を指定せずに varable=だけにします。

```
$ db2set DB2COMM=
```

3-2-4 db2level (DB2 システム・コマンド)

【説明】

現在インストールされている DB2 ユニバーサル・データベースのコードのレベル(FixPak のレベルも含む)に関する詳細な結果を出力します。確認したい環境(サーバー、あるいは、クライアント)にて、db2level コマンドを実行します。FixPak のレベルは、下記サイトにある FixPak番号と照らし合わせることにより、わかります。

V8 FixPak 一覧 :

http://www.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/v8fphist.d2w/report

【使用例】

<AIX 版(64 ビット版)FixPak5 適用済みの出力例>

```
$ db2level
 DB21085I  インスタンス "db2inst1" は、"64" ビットおよび DB2 コード・リリース
"SQL08015" をレベル ID "02060106" で使用します。
情報トークンは、"DB2 v8.1.1.48"、"s040212"、"U496793"、および FixPak "5" です。
製品は "/usr/opt/db2_08_01" にインストールされます。
```

3-2-5 db2ilist (DB2 システム・コマンド)

【説明】

使用可能なインスタンスをすべてリストします。

AIX プラットホームでは、このユーティリティーは /usr/opt/db2_08_01/instance にあります。

【使用例】

```
$ /usr/opt/db2_08_01/instance/db2ilist
db2inst1
```

3-3 注意を要する DB2 の保守を行うツール

3-3-1 db2dart (DB2 システム・コマンド)

【説明】

データベースの保全性が正しいことを検査します。たとえば、このツールを使って以下のことを確認できます。

・制御情報が正しいこと

・データの形式に矛盾がないこと

・データページが正しいサイズであり、含まれている列タイプが正しいこと

・索引が有効であること

このツールはデータベースへの接続が一つも無い状態(オフライン状態)で行わなくてはいけないため、DB2 インスタンスを停止して行ってください。

指定できるオプション、使用上の注意は /H オプションで表示されます。(db2dart /H)

様々なオプションがありますが、Repair Action、 Change State Action に分類されるオプションは、緊急対応手順でのみ、その影響範囲を理解した上で細心の注意を払って使う必要があります。オプションの中にはサポートセンターに問い合わせてパスワードを入手しないと実行できないものもあります。

【使用例】表の保全性を検査します。

SAMPLE データベースの STAFF 表の保全性を検査します。

① あらかじめ STAFF 表の表スペース ID を確認してください。

```
$ db2 "select tableid,tbspaceid from syscat.tables where tabname = 'STAFF'"
TABLEID TBSPACEID
------- ---------
      3         2
  1 record(s) selected.
```

② インスタンスを停止させてください。

```
$ db2stop
```

③ 表スペース ID、表名を指定して d2dart を /T オプションで実行してください。

```
$ db2dart SAMPLE /T /TSI 2 /TN STAFF /SCR M
The requested DB2DART processing has completed successfully!
        Complete DB2DART report found in: SAMPLE.RPT
```

④ 生成されたレポートファイルを確認してください。(上記例では SAMPLE.RPT)

DART

D a t a b a s e A n a l y s i s a n d R e p o r t i n g T o o l

IBM DB2 6000

```
DART (V8.1.0)  Report:
Sat Jun 12 20:35:01 2004 JST

            Database Name: SAMPLE
            Report name: SAMPLE.RPT
            Old report back-up: SAMPLE.BAK
            Database Subdirectory: /home/db2inst1/db2inst1/NODE0000/SQL00001
            Operational Mode: Database Inspection Only (INSPECT)

------------------------------------------------------------------------------
Action option: T
Table-name: STAFF; Tablespace-ID: 2

Connecting to Buffer Pool Services...

      Tablespace file inspection phase start.
      Loading tablespace files.
      Inspecting next tablespace and associated containers.
      Inspecting next tablespace and associated containers.
      Inspecting next tablespace and associated containers.
      Inspecting next tablespace and associated containers.
      Inspecting next tablespace and associated containers.
      Inspecting next tablespace and associated containers.
      6 tablespaces were identified and their containers checked.
      Tablespace file inspection phase end.


      Tablespace-info inspection phase start.
      Tablespace-info inspection phase end.


      Tablespace-info inspection phase start.
      Tablespace-info inspection phase end.

   Table inspection start: DB2INST1.STAFF

      Data inspection phase start. Data obj: 3  In pool: 2
      Data inspection phase end.


      Index inspection phase start. Index obj: 3  In pool: 2
       Scanning pages for index itoken(1) root page:2p.
      Index inspection phase end.

   Table inspection end.

                        ---------------------------------

         The requested DB2DART processing has completed successfully!
                     All operation completed without error;
                  no problems were detected in the database.
```

```
                    Complete DB2DART report found in:
/home/db2inst1/sqllib/db2dump/DART0000/SAMPLE.RPT

_____   D A R T   P R O C E S S I N G   C O M P L E T E   _____
```

3-3-2 INSPECT、db2inspf (DB2 CLP コマンド、DB2 システム・コマンド)

【説明】

INSPECTコマンドは、データベースをオンラインにしたままで、データベースのページの整合性がとれているかどうかを調べることにより、データベースの構造上の保全性を検査します。この検査では、表オブジェクトの構造および表スペースの構造が有効かどうかが調べられます。

db2inspf は、INSPECT CHECK 結果からのデータを ASCII 形式にフォーマットします。 このユーティリティーを使用して、検査の詳細を表示します。 db2inspf ユーティリティーによるフォーマットには、表のみのフォーマット、表スペースのみのフォーマット、エラーのみのフォーマット、警告のみのフォーマット、またはサマリーのみのフォーマットがあります。

【使用例】

SAMPLE データベースの保全性を検査します。

① データベースに接続します。

② INSPECT コマンドを実行します。結果出力ファイルは、診断データ・ディレクトリー・パスに書き込まれます。デフォルトでは、チェック処理によってエラーが検出されない場合、結果出力ファイルは INSPECT 操作の終了時に消去されます。

```
> db2 "inspect check database results INSPECT.DAT"
SQL1141N  操作は完了しましたが、エラーまたは警告がありました。この詳細は、結果フ
ァイル "INSPECT.DAT" にあります。このファイルは db2inspf
ユーティリティーを使ってフォーマットする必要があります。
```

③ エラーが検出された場合は、db2inspf ユーティリティーを実行します。

```
> db2inspf INSPECT.DAT INSPECT.TXT
```

④ エディターを使ってフォーマットされたファイルを表示します。

```
DATABASE: SAMPLE
VERSION : SQL08014
2003-11-25-14.15.41.507000


アクション: CHECK DATABASE
結果ファイル名: INSPECT.DAT

データベース・フェーズが開始しました。

  表スペース・フェーズが開始しました。 表スペース ID: 0
   表スペース名: SYSCATSPACE
   表スペース・タイプ: SMS - System Managed Space、エクステント・サイズ: 32、ページ・サイズ:
    4096、コンテナー数: 1。
   コンテナー名: /database/db2inst1/NODE0000/SQL00002/SQLT0000.0

    表フェーズが開始しました。(ID 符号付き: 2、符号なし: 2; 表スペース ID: 0) : 。
```

```
データ・フェーズが開始しました。 オブジェクト: 2、表スペース: 0。
この表の索引タイプは 2 です。
DAT オブジェクト・サマリー: 合計ページ数 33 - 使用済みページ数 33 - フリー・スペース 7 %。
データ・フェーズが終了しました。

索引フェーズが開始しました。 オブジェクト: 2、表スペース: 0。
 INX オブジェクト・サマリー: 合計ページ数 17 - 使用済みページ数 17。
索引フェーズが終了しました。

LOB フェーズが開始しました。 オブジェクト: 2、表スペース: 0。
 LOB オブジェクト・サマリー: 合計ページ数 320 - 使用済みページ数 289。
 LBA オブジェクト・サマリー: 合計ページ数 2 - 使用済みページ数 2。
LOB フェーズが終了しました。
表フェーズが終了しました。
```

3-3-3 db2untag (DB2 システム・コマンド)

【説明】

何かしらの原因で意図せず残ってしまっている表スペース・コンテナーのタグを強制的に除去します。誤った宛先に対してこのツールを行うとデータベースの損傷につながります。実施する際には、このツールの作用、除去しようとしているコンテナーが確実に他の表スペースで使用していないこと、及び、誤って実行した際の影響度を十分にご理解の上、注意して実行してください。

【使用例】

使用例は2章 3-1 節SQL0294Nを参照ください。

3-3-4 db2iupdt (DB2 システム・コマンド)

FixPak を適用した場合、製品のオプションを導入・除去した場合にインスタンスを更新するために用います。インスタンスは停止して行ってください。FixPak の Readme や製品オプションの導入手順に db2iupdt コマンドの使い方が記載されていることが多いので、その指示に従って実行します。

3-4 OS の稼動状況を確認するための主なツール

本節はAIXに特化した内容のみ記述いたします。

3-4-1 vmstat

【説明】

このコマンドは、仮想メモリの統計情報および CPU 活動に関する統計情報を主に報告します。よく参照する列は、メモリのページイン回数(pi 列)およびページアウト回数(po 列)、および CPU の使用率(us 列+sys 列)、I/O 待ち率(wa 列)です。

ページイン、ページアウトが頻繁に発生していないか、CPU の使用率が異常ではないか、ディスクの I/O 待ちが発生していないか等の情報を得るのに役立ちます。

-t オプションで情報を取得した時間も合わせて表示することが可能です。

また、AIX 5L V5.2 ですと、-v オプションで file memory の設定や現在使用量を確認できます。これは、従来ですと、root 権限が必要な vmtune コマンドで取得する必要がありました。

【使用例】

```
$ vmstat -t 2
```

```
kthr     メモリー             ページ                   フォールト       cpu          時間
----- ----------- ------------------------ ------------ ----------- --------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa hr mi se
 1  1 76077 17656   0   0   0   0    0   0 152  407  98  0  0 99  0 05:32:15
 0  0 76081 17652   0   0   0   0    0   0 156 1488 108  0  0 99  0 05:32:17
 0  0 76081 17652   0   0   0   0    0   0 152 1431 104  0  0 99  0 05:32:19
```

```
$ vmstat -v
```

```
              131072 メモリー・ページ数
              115746 lruable ページ数
               17658 フリー・ページ数
                   1 メモリー・プール数
               24512 ピンされたページ数
                80.1 maxpin パーセント
                20.0 minperm パーセント
                80.0 maxperm パーセント
                28.8 numperm パーセント
               33446 ファイル・ページ数
                 0.0 圧縮されたパーセント
                   0 圧縮されたページ数
                32.2 numclient パーセント
                80.0 maxclient パーセント
               37299 クライアント・ページ数
                   0 スケジュールされたリモート・ページアウト
                   0 pbuf なしでブロック化された保留中ディスク I/O 数
                   0 psbuf なしでブロック化されたページング・スペース I/O 数
                2996 fsbuf なしでブロック化されたファイルシステム I/O 数
```

3-4-2 iostat

【説明】

このコマンドは、入出力活動のモニターと CPU 活動に関する統計情報を主に報告します。システム始動以降のディスクの活動(%tm_act 列)等が表示されます。I/O のバランスが取れているか、特定のディスクに負荷がかかっていないか等の情報を得るのに役立ちます。

-T オプションで、実行時間を取得することができます。

【使用例】

```
$ iostat -d -T 2
```

```
Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn    時間
hdisk0            0.0      0.0      0.0          0         0  05:40:55
hdisk3            0.0      0.0      0.0          0         0  05:40:55
hdisk2            0.0      0.0      0.0          0         0  05:40:55
hdisk1            2.5     16.0      4.0          0        32  05:40:55
cd0               0.0      0.0      0.0          0         0  05:40:55
```

3-4-3 sar

【説明】

オペレーティング・システム内に累積された活動状況カウンターの内容を報告します。主に CPU の使用状況収集と、収集したデータの処理および表示に使用します。サンプリングの頻度を高くすると、システム負荷が増加して、パフォーマンス上の問題を悪化させる可能性がありますのでご注意ください。

オプション

-o バイナリデータを出力します。

-u CPU 使用率を出力します。

【使用例】

① コマンドをバックグラウンドで実行し、60 秒間隔で 10 回システム活動データを収集して、temp ファイルに保管します。

```
$ sar  -o  temp  60  10
```

② temp ファイルから CPU 情報を抽出し、書式化されたレポートを標準出力に出力します。

```
$ sar  -u  -f  temp
```

3-4-4 ps

【説明】

プロセスの現在の状況を表示します。 アクティブ・プロセスの現行状況、およびそれに関連するカーネル・スレッドを、標準出力に書き出します。

オプション

-f 完全なリストを作成します。

-e 現在実行中の全てのプロセスに関する情報を表示します。

【使用例】

2 章 2-3-3-1 節DB2 プロセスの稼働の確認を参照してください。

3-4-5 netstat

【説明】

ネットワークの利用状況を表示します。

ソケットの利用状況やルーティング、ネットワーク・インターフェース等、ネットワークに関する情報を得ることができます。

オプション

-c ネットワーク・バッファー・キャッシュの統計情報を表示します。

-i すべての構成されたネットワークインタフェースの状態を表示します。

【使用例】

2 章 2-2-2-3 節N/W状態確認を参照してください。

3-4-6 iptrace

【説明】

インターネット・プロトコルに対して、インターフェース・レベルのパケット・トレースを行います。

すべてのインターフェース上で、任意のホストとの間で送受信されるパケットを記録するには、次のフォー

マットでコマンドを入力します。

【使用例】以下の例では、10秒間 iptrace を取得して、結果をテキスト形式にフォーマットして表示します。

```
$ startsrc -s iptrace -a "/tmp/nettrace" ; sleep 10 ; stopsrc -s iptrace
$ ipreport /tmp/nettrace
```

3-4-7 lslv

論理ボリュームに関する情報を表示します。*LogicalVolume* の特性と状況を表示するか、*PhysicalVolume* 上の物理区画用の論理ボリューム割り当てマップをリストします。

3-4-8 lsvg

ボリューム・グループに関する情報を表示します。*VolumeGroup* パラメーターを使うと、ボリューム・グループに関する情報だけが表示されます。*VolumeGroup* パラメーターを使わなければ、定義済みボリューム・グループすべての名前のリストが表示されます。

3-4-9 df

【説明】

ファイルシステム上の合計スペースと使用可能なスペースの情報を表示します。

【使用例】

2 章 2-3-2-5 節ファイルシステム空き容量確認を参照してください。

3-4-10 kill –36

【説明】

AIX では kill コマンドでプロセスへのシグナル送信が可能です。シグナルを誤って送信してしまうとプロセスの終了を伴う場合がございますので、十分注意が必要です。

/usr/include/sys/signal.h にｼｸﾞﾅﾙが定義されておりますが、36 はプログラムの例外のシグナルです。

DB2 プロセスがプログラム例外のシグナルを受け取ると、データベースマネージャー構成パラメーターの DIAGPATH で指定されたパスにトラップ・ファイルを出力します。トラップ・ファイルには、実行された最後のステップの関数の流れが含まれています(2-1-4 節 DB2 トラップ・ファイル参照)。よって、このコマンドを間隔を開けて数回実行することにより、DB2 プロセスが停止しているか、処理中であるかの判断を行うことが可能です。

プロセスがハングしていると判断された場合に、サポートセンターより資料取得の依頼をされることもあります。

【使用例】

以下に db2sysc に異常の可能性があった場合に稼働状況を確認する例を記載します。

※正常時には絶対に実行しないで下さい。

①db2syscのプロセスの PID を確認する。

```
$ ps  -ef  | grep  db2sysc
```

②kill コマンド実行する

```
$ kill  -36  < db2sysc_pid>
```

③ トラップ・ファイルの存在確認

データベースマネージャー構成パラメーターの DIAGPATH で指定されたパスにトラップ・ファイルが作成されます。ファイル名の最初の文字は”t”で、その後にプロセス ID(pid)が続きます。

3-4-11 topas / nmon

【説明】

実際にシステムをモニタリングする場合には、cpu, i/o, memory, network についてまとめて確認できると便利です。topas あるいは nmon を使うと可能になります。以下のような情報を取得できます。なお、nmon の方は、画面出力の結果を記録することも可能で、以下のサイトからダウンロードして使用してください。

http://www-106.ibm.com/developerworks/eserver/articles/analyze_aix/

```
Topas Monitor for host:    dmaix2                EVENTS/QUEUES    FILE/TTY
Sun Jun 13 05:47:50 2004   Interval:  2          Cswitch    1146  Readch   398.4K
                                                 Syscall    2048  Writech 1002.6K
Kernel    4.5   |##                          |  Reads        62  Rawin         0
User     16.0   |#####                       |  Writes       82  Ttyout       56
Wait     62.0   |##################          |  Forks         0  Igets        0
Idle     17.5   |######                      |  Execs         0  Namei        57
                                                 Runqueue   0.0  Dirblk       0
Network  KBPS   I-Pack  O-Pack   KB-In  KB-Out  Waitqueue  1.0
en0       0.2     4.0     3.0     0.2     0.3
lo0       0.0     0.0     0.0     0.0     0.0  PAGING           MEMORY
                                                 Faults     612  Real,MB     511
Disk    Busy%     KBPS     TPS KB-Read KB-Writ  Steals       0  % Comp     64.3
hdisk1   73.0   1250.0   126.5   464.0  2036.0  PgspIn       0  % Noncomp  26.7
hdisk0    0.0      0.0     0.0     0.0     0.0  PgspOut      0  % Client   29.7
hdisk2    0.0      0.0     0.0     0.0     0.0  PageIn      57
cd0       0.0      0.0     0.0     0.0     0.0  PageOut    252  PAGING SPACE
hdisk3    0.0      0.0     0.0     0.0     0.0  Sios       310  Size,MB    1536
                                                                 % Used       0.7
Name            PID  CPU%  PgSp Owner           NFS (calls/sec)  % Free      99.2
db2sysc      335942  17.5   2.1 db2inst         ServerV2       0
topas        380990   0.5   1.2 root            ClientV2       0   Press:
db2sysc      270516   0.5   0.4 db2inst         ServerV3       0   "h" for help
dtgreet       77950   0.0   2.1 root            ClientV3       0   "q" to quit
gil           40980   0.0   0.1 root
i4llmd       262276   0.0   0.8 root
rpc.lockd    196720   0.0   0.2 root
nfsd         204904   0.0   0.2 root
syncd        106586   0.0   0.6 root
```

3-4-12 svmon

このコマンドにより仮想メモリーの状況を取得できます。

db2 エージェントのプライベート・メモリー量は、db2 メモリー・トラッカーで取得できるが、実際には、OS では指定サイズよりも大きいメモリーを確保している場合があるので、実使用量を正確に知る必要がある場合には、svmon を使用する。

このコマンドの実行には、root 権限が必要です。

```
$ svmon -P <process_id>
```

```
$ svmon -U <db2_instance_owner>
```

のように使用します。

詳しくは、AIX のマニュアルを参照してください。

3-4-13 vmtune

JFS / JFS2 の file memory に関する設定および現在使用量を確認します。

使用できる file memory のサイズを不用意に大きくしておくと、DB2 用のメモリーを圧迫し、ページングが発生する場合があります。デフォルトの値はリアルメモリーの約 80%まで file memory が使用できるようになっていますので、一般的な DB2 システムにとっては大きく、通常は調整が必要です。

JFS の場合、minperm, maxperm の値を調整し、strict_maxperm=1 と指定します。

JFS2 の場合には、maxclient の値を調整します。

さらに、大規模なデータベースの場合の対象の sequential read / write に対して、ファイル・キャッシュの有効利用のために、ファイル・キャッシュの使用を必要最小限に抑えたい場合があります。そのためには、ファイル・システムの mount オプションで、Release Behind(-o rbr/rbw/rbrw)オプションを指定することを検討してください。

また、データのロードの際には、入力ファイルの読取り部分がボトルネックになる場合があります。その場合には、ファイル・システムの先読みページ数の指定を大きくするように変更すると効果がある場合があります。JFS の場合、maxpageahead、JFS2 の場合、j2_maxPageReadAhead を調整します。

このコマンドの実行には、root 権限が必要です。

なお、このコマンドは、bos.adt.samples というファイルセットに入っています。

詳しくは、AIX のマニュアルを参照してください。

3-4-14 その他トレース、情報収集ツール (truss, trace, filemon, perfpmr)

その他、プロセスで発行されたシステム・コールをトレースする truss, システム全般(システム・コール、ファイル I/O, 物理 I/O, デバイス・ドライバーなど)をトレースする trace (smitty trace)、ファイル・システムのパフォーマンスをモニターする filemon, パフォーマンスに関して網羅的に情報を収集する perfpmr などがあります。perfpmr については、以下のサイトからダウンロードしてください。

ftp://ftp.software.ibm.com/aix/tools/perftools/perfpmr/

4 データ保守／アプリケーション保守用ツールのご紹介

ここで紹介するコマンドの詳細については、『コマンド解説書』を参照してください。

4-1 EXPORT　(DB2 CLP コマンド)

【説明】

表からファイルにデータを抽出するために使用します。出力できる形式は主に以下の2種類です。ASC形式には Export できません。

DEL 形式:

delimiter 形式（所謂CSV形式）。文字型データは”（二重引用符）で括られ、数値型データの前後には何も付きません。また、列間は,(コンマ)で仕切られます。これらの文字区切り文字”および列区切り文字,については、文字を変更することも可能です。

IXF 形式:

DB2 特有の形式。基本的にはデータ変換ができるだけ起こらないようにデータが格納されていて、一般的には、性能的にも出力ファイルのサイズの面でも有利です。また、表の定義情報も含んでいるため、データの移動先で、表を新規に作成しながら、データを import することも可能です(load には表を作成する機能はありません)。また、コードページ情報も含んでいるため、別のシステムにもって行って、別のコードページのデータベースに Load/Import することも可能になっています。

<オプション>

- MODIFIED BY CHARDEL, MODIFIED BY COLDEL オプション(DEL 形式にのみ有効)

この指定により、文字区切り文字および列区切り文字を変更できます。modified by chardel| coldel; のように、chardel, coldel の後ろすぐにその区切り文字を指定します。’のようにオペレーティングシステムで特別な意味を持つ文字の場合には、2つ続けて書くようにしてます。たとえば、modified by chardel’’ のようにします。区切り文字として扱える文字には、DBSC の場合、0x40 よりも大きくできない制限があります。その他詳しくは、『コマンド解説書』の Export の項を参照してください。なお、区切り文字は、16進表示でも指定できます。たとえば、列区切り文字にタブ(0x09)を指定したい場合には、modified by coldel0x09 と指定します。

- LOBFILE パラメーター、LOBS TO パラメーター、MODIFIED BY LOBSINFILE オプション

LOBデータがある場合には、LOB データをファイルに出力することができます。そのために上記3つの指定を行います。LOBFILE lobdata と指定すると、出力されるごとに、lobdata.001, lobdata.002, lobdata.003 ….と自動的に名前がつけられます。さらに、LOBS TO /lobpath と指定すると、/lobpath/lobdata.001, /lobpath/loobdata.002, …に順次出力されて行きます。また、export file の該当個所には、そのファイル名が出力されるようになっていて（path 名は出力されません）、後続の処理で、再度 import あるいは load できるようになっています。また、LOB データが 999 個以上存在する場合には、LOBFILE パラメーターに複数ファイル名を指定するか、LOBS TO パラメーターで出力先 path を複数指定するようにしてください。1つの path の指定につき、LOBFILE で指定されたファイル名が999個 (*.001 - *.999) ずつ順次使用されて行きます。

【使用例】

<使用例 1>

表 test にあるデータを、test.csv ファイルにエクスポートします。

```
$ db2  “export  to  test.csv   of   del  select  *  form   test”
   SQL3104N  エクスポート／アンロード・ユーティリティーが、ファ イル"test.csv" へのデータのエクスポート／アンロードを開始しています。
   SQL3105N  エクスポート／アンロード・ユーティリティーが、 "42094"行のエクスポート／アンロードを終了しました。
```

<使用例 2>

LOB を含む表（emp_photo）にあるデータを empno 順にソートしてエクスポートします。LOB データはフ

ァイルに出力します。

```
$ db2 “export to photo.del of del lobs to ./path1, ./path2 lobfile lob01, lob02
modified by lobsinfile chardel’’ coldel; messages exp_photo.log
select * from emp_photo order by empno”
エクスポートされた行数: 12
```

exp_photo.log ファイルには以下が記録されます。

```
SQL3104N エクスポート・ユーティリティーが、ファイル "photo.del" へのデータの
エクスポートを開始しています。
SQL3105N エクスポート・ユーティリティーが、"12"行のエクスポートを終了しました。
```

4-2 IMPORT (DB2 CLP コマンド)

【説明】

DEL 形式(CSV形式)、IXF 形式、ASC 形式(固定長レコード)のデータを表に挿入します。IXF 形式の場合には、入力ファイル内のテキスト・データのコードページと挿入先のデータベースのコードページが異なっていても、データベースのコードページに自動的に変換して挿入します。DEL 形式、ASC 形式の場合で、コード変換が必要な場合には、LOADコマンドで modified by codepage=xxx と入力ファイルのコードページを指定してロードすることで対応できます。import では、modified by codepage=xxx は使用できません。

<オプション>

- MODIFIED BY CHARDEL, MODIFIED BY COLDEL オプション

  EXPORT と同様です。

- LOBS TO パラメーター、MODIFIED BY LOBSINFILE オプション

  EXPORT で LOB データをファイルに取り出した場合、そこから、同様に IMPORT にて表にデータを挿入できます。なお、IMPORT には、LOBFILE パラメーターはありません。

- COMMITCOUNT パラメーター

  大量のデータを import する場合に、ログ・サイズが小さいとログ・フルを引き起こしてしまいます。また、一切コミットせずに import するのも性能上不利です。そこで、このパラメーターで、何行に一回 commit を発行するかを指定してログ・フルや性能の劣化を防ぎます。1000から10000を指定することが多いようです。

- その他 MODIFIED BY xxx オプション

数多くのオプションがありますので、『コマンド解説書』を参照ください。

【使用例】

<使用例 1>user.csv ファイルのデータを DB2 データベース内の表 USER にインポートする。

```
$ db2 “import from user.csv of del insert into user”
    SQL3109N ユーティリティーが, ファイル "user.csv" からデータのロードを開始しています。
    SQL3110N ユーティリティーが処理を完了しました。"5392"行が入力ファイルから読み取られました。
    SQL3221W ...COMMIT WORK が開始されました。入力レコード数 = "5392"。
    SQL3222W ...すべてのデータベース変更のコミットが成功しました。
    SQL3149N "5392"行が、入力ファイルから処理されました。 "5392" 行が, 正常に表に挿入されました。
    0 行が, 拒否されました。
```

```
読み込まれた行数 = 5392
スキップされた行数 = 0
挿入された行数 = 5392
更新された行数 = 0
拒否された行数 = 0
コミットされた行数 = 5392
```

<使用例2>LOBを含む表(emp_photo)にデータをインポートする

```
$ db2 “import from photo.del of del lobs from ./path1/, ./path2/ modified by lobsinfile chardel’’
coldel; commitcount 5 messages imp_photo.log replace into emp_photo”
  読み込まれた行数        = 12
  スキップされた行数      = 0
  挿入された行数          = 12
  更新された行数          = 0
  拒否された行数          = 0
  コミットされた行数      = 12
  imp_photo.log には以下が記録されており、５行に一回 commit が実行されたこともわかります。
  SQL3109N  ユーティリティーが、ファイル "photo.del"からデータのロードを開始しています。
  SQL3221W  ...COMMIT WORK が開始されました。入力レコード数 = "5"。
  SQL3222W  ...すべてのデータベース変更のコミットが成功しました。
  SQL3221W  ...COMMIT WORK が開始されました。入力レコード数 = "10"。
  SQL3222W  ...すべてのデータベース変更のコミットが成功しました。
  SQL3110N  ユーティリティーが処理を完了しました。"12"行が、入力ファイルから読み取られました。
  SQL3221W  ...COMMIT WORK が開始されました。入力レコード数 = "12"。
  SQL3222W  ...すべてのデータベース変更のコミットが成功しました。
  SQL3149N  "12" 行が、入力ファイルから処理されました。 "12"行が、正常に表に挿入されました。
  "0" 行が、拒否されました。
```

<使用例 3>LOBを含む表(emp_photo2)を新規に作成してデータをインポートする

```
$ db2 “import from photo.ixf of ixf lobs from ./path1/, ./path2/ modified by lobsinfile
  messages imp_photo.log create into emp_photo2 in data_ts index in index_ts long in long_ts”
  読み込まれた行数        = 12
  スキップされた行数      = 0
  挿入された行数          = 12
  更新された行数          = 0
  拒否された行数          = 0
  コミットされた行数      = 12
  imp_photo.log には以下が記録されます。
  SQL3150N  PC/IXF 形式ファイルの H レコードには、製品 "DB2     02.00"、日付
  "20020629"、および時刻 "003216" が入っています。
  SQL3153N  PC/IXF 形式ファイルの T レコードは、名前 "photo.ixf"、修飾子
  ""、およびソース "            " を持っています。
  SQL3109N  ユーティリティーが、ファイル "photo.ixf"からデータのロードを開始しています。
  SQL3110N  ユーティリティーが処理を完了しました。"12"行が、入力ファイルから読み取られました。
  SQL3221W  ...COMMIT WORK が開始されました。入力レコード数 = "12"。
  SQL3222W  ...すべてのデータベース変更のコミットが成功しました。
  SQL3149N  "12" 行が、入力ファイルから処理されました。 "12"
  行が、正常に表に挿入されました。"0" 行が、拒否されました。
```

4-3 LOAD (DB2 CLP コマンド)

【説明】

IMPORT コマンドよりも高速にデータを表に挿入でき、初期データロードなどの、大量データの移動に適しています。(詳細は『コマンド解説書』参照)

<回復に関する考慮事項>

ロード処理中に、変更内容のログ記録は行われないため高速です。そのかわり、デフォルトのロードオプションではRESTOREコマンドとROLLFORWARDコマンドを使用して、ロード操作のロールフォワード回復を実行することはできません。

LOAD コマンドにはパフォーマンスと回復可能性のバランスを取る為に下記の 3 つのオプション設定が用意されています。

- Copy　No オプション　(デフォルト設定)

このオプションが指定されていて、なおかつ保存ログ方式の場合、ロード表が存在する表スペースはロード後にバックアップ保留状態に置かれ、表スペースはデータベースまたは表スペースのバックアップが取得取得されるまでは更新アクセスが不能になります。

<使用例>IXF ファイルの内容を既存の表 SHOP.TOOLS に挿入する。

```
$ db2  “load  from  tool.ixf  of  ixf  massages load.msg  insert  into  shop.tools
  for exception  shop.badtools copy no
```

(load.msg:ロード中に生成されたメッセージの保存場所　※問題判別のために必須)

(shop.badtools:主キーや固有索引制約に違反した行の移動先)

- Copy　Yes オプション

このオプションが指定されていて、なおかつ保存ログ方式の場合、ロード処理によって行われた変更内容のコピーが保管され、ロード後に表スペースはバックアップ保留状態のままにはなりません。但しこのオプションは循環ログを方式のデータベースには適用できません。ロード操作をロールフォワード回復の対象にさせたい場合にこのオプションを使います。

<使用例>表 SHOP.TOOLS の内容を置き換え、表とその索引に関する完全な統計情報を収集する。

```
$ db2  “load  from  tool.ixf  of  ixf massages  load.msg  replace  lnto  shop.tools
  statistics yes  indexes  all  copy  y e s  to <copy-file-path>”
```

(load.msg:ロード中に生成されたメッセージの保存場所　※問題判別のために必須)

- Nonrecoverable オプション

このオプションを使用すると、表のロード後に表スペースはバックアップ保留状態にならず、またデータのコピーも作成されなくなります。したがってユーザーは、ロードが完了したらすぐに表と新しいデータにアクセスできます。ただしロード後の表データに対する変更内容はログには記録されないので、データベースを以前のバックアップ・イメージより復元し、ロード操作時点を過ぎてロールフォワード回復するとその表は無効になってしまいます

このオプションは大きな読み取り専用のデータをロードする場合に便利です。

<使用例>IXF ファイルの内容を既存の表 SHOP.TOOLS に挿入する。

```
$ db2  “load  from  tool.ixf  of  ixf  massages load.msg  insert  into  shop.tools
  for exception  shop.badtools nonrecoverable
```

(load.msg:ロード中に生成されたメッセージの保存場所　※問題判別のために必須)

(shop.badtools:主キーや固有索引制約に違反した行の移動先)

<ロードの保留時の解除方法>

表スペースはロード・フェーズと作成フェーズ中にこの状態になります。これら2つのフェーズのどちらかの間にロード操作が失敗あるいは、ユーザーの操作により FORCE した場合、表スペースはロード保留状態のままになります。

ロード操作が失敗した場合は、ロードのメッセージ・ファイルやエラー・ログを調べて失敗の理由を判別する必要があります。問題を判別して修正した後、ユーザーは以下のいずれかの方法で再始動できます。

① LOAD コマンドに INSERT オプションを使用していた場合

<説明>

メッセージファイルに示されている問題を訂正し、INSERT オプションを RESTART オプションに置き換えて LOAD コマンドを再起動します。RESTART オプションで再起動した場合、ロード操作は失敗したフェーズから再始動されるか、ロード・フェーズで失敗して LOAD コマンドに SAVECOUNT オプションが指定されていた場合には、最後の一貫性ポイントから再始動されます。

<使用例>

1.以下の INSERT オプションを使用したロード操作が失敗しロード保留状態になったとします。

$ db2 “load from 〈*file-name*〉 of 〈*file-type*〉 insert into 〈*table-neme*〉”

2. INSERT オプションを RESTART オプションに置き換えて再起動させてください。

$ db2 “load from 〈*file-name*〉 of 〈*file-type*〉 restart into 〈*table-neme*〉”

※ 空の表へのロードの場合は、RESTART オプションの代わりに REPLACE をオプションを用いることもできます。

② LOAD コマンドに REPLACE オプションを使用していた場合

<説明>

メッセージファイルに示されている問題を訂正し、再び REPLACE オプションを指定して LOAD コマンドを再起動します。

<使用例>

1.以下の REPLACE オプションを使用したロード操作が失敗しロード保留状態になったとします。

$ db2 “load from 〈*file-name*〉 of 〈*file-type*〉 replace into 〈*table-neme*〉”

2.再び REPLACE オプションで起動させてください。

$ db2 “load from 〈*file-name*〉 of 〈*file-type*〉 replace into 〈*table-neme*〉”

③ 上記で問題が解決できない場合(TERMINATE オプション)

<説明>

上記の方法で問題が解決できない場合は、TERMINATE オプションを使用してロード操作を終了し、ロード操作が開始された時点まで操作をロールバックできます。途中に一貫性ポイントがあっても通過します。操作に関係する表スペースの状態は正常に戻され、すべての表オブジェクトが一貫性のある状態に保たれます (索引の再作成が次回のアクセスで自動的に行われることになっている場合は、索引オブジェクトは無効としてマークされます)。

LOAD コマンドが最初に REPLACE オプションを指定して起動された場合は、その起動前に表

に存在していたデータは全て失われ、表は空のままになります。INSERT オプションを指定して起動された場合は、新しいデータが表から削除され、古いデータは影響を受けず表は LOAD コマンドを起動する前の状態に戻ります。

<使用例>

```
$ db2 “load from <file-name> of <file-type> terminate into <table-neme>”
```

4-4 REORG (DB2 CLP コマンド)

【説明】

レコードの追加、更新、削除により以下のような現象が発生して性能劣化につながる場合があります。

・ページ・オーバーフローが発生して、ディスクI/O効率全般が悪くなる。

・あるインデックスの順にデータレコードが配置されていて(あるいは、クラスターインデックスを使用している)効率良いディスクI/O処理ができていたのが、データレコードの配置順が乱れてしまい、ディスクI/O効率が悪くなる。

これらの問題を解消するために REORGコマンドを使用します。REORGコマンドにより、以下のことが実現できます。

・指定したインデックス順にデータを並び替えます。

・オーバーフロー・ページをなくし、指定されたフリースペースを設けながらデータを再編成し直します。このフリースペースにより、ページ・オーバーフローや並び順が乱れることをできるだけ防止します。

再編成は表単位で行います。再編成の実行では、ワーク用に一時的に多くの表スペースを使用します。インデックス順に並び替える場合には、特に多くの領域を必要としますので、REORG実行時に作業専用の一時表スペースを作成することを検討してください。その際はREORG対象の表が属している表スペースのページサイズと同じページサイズで一時表スペースを作成する必要があることに注意してください。また通常は一時表スペースはREORGを行う表のデータ量の2～3倍必要です。なお、REORGでは、データを再編成したあとに続いてインデックスの再作成を内部的に行います。

DB2 V8からは、オンラインで表を再編成することも可能になり、以下の2方式が選択可能になりました。

1. インプレース・アプローチ(allow read/write access)

【説明】

INPLACEオプションを使用することで、再編成中の表へ読み取り/書き込みアクセスが可能になります。また、表スペース内だけで再編成が行われるため、一時作業領域の容量が少なくてすみます。再編成中の中断、再開、強制終了が可能です。

【使用例】

table1 表のインプレース再編成を開始、一時停止、および再開する例です。

```
> db2 reorg table table1 index ind1 inplace allow write access start
> db2 reorg table table1 inplace pause
> db2 reorg table table1 inplace resume
```

2. シャドウ・コピー・アプローチ(allow read/no access)

【説明】

再編成のために大きな一時表スペースが必要となりますが、インプレース・アプローチと比較して、再編成は速くなります。

【使用例】

作業域としてシステム一時表スペース TEMPSPACE1 を使用して、指定したインデックスの順で 表を再編成する例です。

他のユーザーがその表に対して、読み取り専用のアクセスを行うことができることを指定します。

```
> db2 reorg table table2 index ind2 allow read access use tempspace1
```

DB2 V7のように、再編成される間、他のユーザーがその表にアクセスできないことを指定します。

```
> db2 reorg table table3 index ind3 allow no access use tempspace1
```

4-5 REORGCHK (DB2 CLP コマンド)

【説明】

レコードの削除や、追加により、索引順のデータレコードの並びが乱れた場合や、追加レコードのため REORG を実行する必要があるかどうかについて、統計情報を元に検査し報告します。実行時に最新の統計情報を収集することもできます。

どのような報告がされるかについては、『コマンド解説書』を参照してください。

【使用例】

<使用例 1>現在接続しているデータベースの STAFF 表に対して、まず、統計情報を再収集してから、検査を実行する

```
$ db2 reorgchk update statistics on table db2inst1.staff
```

<使用例 2>現在接続しているデータベースのすべての表に対して、統計情報を再収集せずに検査を実行する。

```
$ db2 reorgchk current statistics on table all
```

4-6 RUNSTATS (DB2 CLP コマンド)

【説明】

表（および列）と索引の物理的な統計を収集しカタログ表を更新します。収集される情報は、レコード数、ページ数、並び順(cluster ratio)、カーディナリティなどです。一般的に、前回RUNSTATSを実行してから統計情報が大きく変わったもの対して、および、オブジェクトに変更があったものに対して取得します。通常、以下のような操作後に実行します。

- バッチ処理による大量更新・挿入
- 再編成（REORG）
- 新しい索引の作成
- 列の追加

統計情報が変わると、SQLの最適なアクセスパスも変わりますので、RUNSTATS後には、静的SQLはパッケージを再度バインドして、アクセスパスを最適なものに更新します。動的SQLを利用している場合には、統計情報を実行時に確認するため再バインドは不要です。

【使用例】

<使用例1>可能なすべての統計 (分布および拡張索引) を収集 (通常、この指定が推奨)

```
$ db2  runstats  on  table  smith.table1  with  distribution  and  detailed  indexes all
```

<使用例2>索引のみの基本統計を収集

```
$ db2  runstats  on  table  smith.table1  for  indexes  all
```

4-7 BIND (DB2 CLP コマンド)

【説明】

ソース・プログラムの事前コンパイルで生成されたバインド・ファイルよりパッケージを生成してそれをデータベースに保管します。その際、静的なSQLについては、アクセスパスも生成されて保管されます。

【使用例】

myapp.bnd (myapp.sqc プログラムのプリコンパイル時に生成されるバインド・ファイル) をバインドします。オプションとして、検索について結果セットがブロック化されるようにし、全ユーザーにパッケージの実行権限を付与するよう指定します。

```
$ db2  bind  myapp.bnd  blocking  all  grant  public
```

4-8 REBIND (DB2 CLP コマンド)

【説明】

RUNSTATS実行後、静的SQLのアクセスパスを最新の統計情報を元に最適なものに更新するために、REBINDコマンドを実行します。このコマンドは、バインド・ファイルを必要としません。

パッケージ名は、デフォルトでは、アプリケーション・プログラムのソースファイル名（ファイル拡張子は含まない）で、最大8文字となります。

【使用例】

ある表に対して新しい索引を作成後、この索引をアプリケーション(ソースファイル：db2cert.sqc)が使用できるようにリバインドします。

```
$ db2  rebind  db2cert
```

4-9 db2rbind (DB2 システム・コマンド)

【説明】

このコマンドは、データベース内のすべてのパッケージのリバインドを一括して行うためのものです。特定のパッケージのみリバインドしたい場合は db2rbind ではなく REBIND か BIND コマンドを使用します。

【使用例】 データベース名“samle”にリバインドを実施した例（ragi.txt はログファイル）

①リバインド実施

```
$ db2rbind sample -l ragi.txt
```

データベース 'SAMPLE' の再バインドが正常に完了しました。

（バインド実施後正常に完了すると、上記のメッセージが出力します）

②バインドの戻り値確認

```
$ echo $?
100
```

(バインド成功時”100”が表示、データ・ベースがない場合“0”が出力されます)

③ログファイルの確認

```
$ cat ragi.txt
  cat: 0652-050 ragi.txt をオープンできません。
```

（バインド成功時は、ログファイルは作成されません）

4-10 db2look　(DB2 システム・コマンド)

【説明】

データベースまたはデータベース・オブジェクトを再作成するために必要な DDL や統計情報など各種情報を抽出します。実働データベースの模擬環境を構築する際に役立ちます。またテスト・データベースの登録変数および構成パラメーター設定を、実動データベースの設定に適合させることにも役立ちます。

テスト・データベースのカタログ統計と実動データベースのカタログ統計を一致させる更新ステートメントを生成するには、実動データベースに対して-m オプションを使用します。データベース・カタログから 1 つまたは複数の表用の DDL を生成するには-e オプションを使用します。

db2look の初歩的な情報を知りたい場合には、db2look と入力してください。詳細については、db2look –h と入力してください。

オプション

-a　すべてのユーザーによって作成されたすべてのオブジェクトが対象になります。 たとえば、このオプションと -e オプションが共に指定される場合、 データベース内のすべてのオブジェクト用の DDL ステートメントが抽出されます。 このオプションと -m オプションが共に指定される場合、 データベース内のすべてのユーザー作成表および索引用の UPDATE 統計ステートメントが抽出されます。

-e　データベース・オブジェクト用の DDL ステートメントを抽出します。 このオプションは、-m オプションと一緒に使用できます。

-m　表、列、および索引についての統計を複製するために必要な UPDATE ステートメントを生成します。

【使用例】

①データベース DEPARTMENT でユーザー walid によって作成されたオブジェクト用の DDL ステートメントを生成します。 db2look の出力は、以下のようにしてファイル db2look.sql に送信します。

```
$ db2look -d department -u walid -e -o db2look.sql
```

②データベース DEPARTMENT でユーザー walid よって作成された表および索引の統計を複製するための UPDATE ステートメントを生成します。

```
$ db2look -d department -u walid -m -o db2look.sql
```

③データベース DEPARTMENT ですべてのユーザーによって作成されたオブジェクトの DDL ステートメントを生成します。

```
$ db2look -d department -a -e -o db2look.sql
```

5 DB2 インスタンスの強制終了方法

通常のインスタンス終了方法は、接続が一つもないことを確認(list application コマンド)の上、db2stop コマンド(オプションなし)で終了させます。 強制的にすぐ DB2 インスタンスを停止したい場合には、以下の順番で試してください。本節はAIXに特化した内容を記述いたします。

5-1-1 force オプション付きの db2stop コマンド(db2stop force)

【説明】

接続中のアプリケーションを全て強制切断後に、インスタンスを停止させます。

【使用例】

```
$ db2stop force
SQL1064N  データベース･マネージャーの停止処理が正常に終了しました。
```

5-1-2 db2_kill コマンド (ESE) / db2nkill コマンド (WSE)

db2_killコマンドは、ESE(エンタープライズ・サーバー・エディション) 向けのもので、複数のデータベース･パーティション･サーバーで実行されているすべてのプロセスを強制的に停止し、すべてのデータベース・パーティション・サーバーのすべてのリソースを終結処理します。このコマンドは、データベースを(一時的に)不整合にしますので、コマンド実行後に再度データベースを使用する前に、クラッシュ・リカバリーを実行する必要があります。このコマンドは IBM サービスからの指示による以外は発行しないでください。

db2nkill コマンドは、WSE (ワークグループ・サーバー・エディション) において、db2_killと同様の目的で利用されるものです。起動は、

> db2nkill 0

とします。

なお、db2_kill と同様に、IBMサービスからの指示による以外は発行しないでください。

V７では、db2stop -kill というコマンドがありましたが、V8 ではサポートされていません。

なお、WSEの製品レベルによっては、db2nkill がパッケージされておりません(APAR IY46337)。

必要な場合、IBM サービスから入手してください。

5-1-3 kill コマンド

これは、緊急時の最後の手段とお考えください。killコマンドを使用して、停止できないdb2 エージェントを終了させます。それでも終了しなければdb2syscプロセスを終了させます。この後、5-1-5 強制終了後のIPCリソースのクリーンアップ－ipclean, ipcs, ipcrmコマンドの手順に従い全ての関連リソースが削除されたことを確認します。

5-1-4 強制終了後のIPCリソースのクリーンアップ－ipclean, ipcs, ipcrm コマンド

【説明】

DB2 インスタンスを強制終了させた場合は、関連するリソースがクリーンアップされずに残っている可能性があります。 その確認、及び、残っている場合はクリーンアップを行います。特に kill コマンドにてプロセスを強制終了した後には必ず行ってください。

【使用例】

① まず DB2 関連プロセスで停止していないものがないか確認。

```
$ ps -ef  | grep <instance-name> | grep db2 | grep -v grep
```

※ DB2 関連のプロセスは db2 から始まる名前になりますので、これが残っていないことを確認します。

② ipcs コマンドで残っている IPC リソースを確認。

```
$ ipcs | grep <instance-name>
q  41287855   00000000 -Rrw-------    <instance-name>  sysadm
q  18874571 0x71056083 --rw-rw--w-    <instance-name>  sysadm
q  73793746   00000000 --rw-------    <instance-name>  sysadm
q   9306325 0x01ebdff1 --rw-------    <instance-name>  sysadm
q   4325625 0x72056083 -Rrw-rw--w-    <instance-name>  sysadm
q   5243135 0x77056083 --rw-rw-r--    <instance-name>  sysadm
q   7864579 0x01db171a --rw-------    <instance-name>  sysadm
m  11927638 0x74056083 --rw-rw-rw-    <instance-name>  sysadm
m    917594 0x61056083 --rw-------    <instance-name>  sysadm
s  35651743 0x74056083 --ra-ra-ra-    <instance-name>  sysadm
s  57016554   00000000 --ra-ra----    <instance-name>  sysadm
s  35783019 0x73056083 --ra-ra-ra-    <instance-name>  sysadm
s   1048953   00000000 --ra-ra----    <instance-name>  sysadm
s   1048954   00000000 --ra-ra----    <instance-name>  sysadm
```

③ ipclean コマンドで DB2 の IPC リソースをクリーンアップする。必ずインスタンスオーナーで実行。

```
$ ipclean
ipclean: Removing DB2 engine and client's IPC resources for <instance-name>.
```

④ ②の手順を繰返し、再度残っている IPC リソースを確認。

```
$ ipcs | grep <instance-name>
q   9306325 0x01ebdff1 --rw-------    <instance-name>  sysadm
q   7864579 0x01db171a --rw-------    <instance-name>  sysadm
```

※ 残っている場合は、DB2 の IPC リソースではなく、<*instance-name*> でログインしている別のアプリケーションの IPC リソースの可能性もあります。 DB2 の IPCリソースであれば④の手順に進み再度クリーンアップします。

⑤ ③の手順でもクリーンアップできなかったリソースを ipcrm コマンドで削除。

```
$ ipcrm -q <msqid>
```

※ 削除したいリソースが message queue ならば -q、shared memory ならば -m、 semaphore ならば -s のオプションを使います。

2章　障害時の対応方法

1 障害時の履歴および環境の保存

まず問題判別をする際、報告された問題を識別または解決するために取った処置の履歴は必ず保存してください。後の解析時、あるいは同様の問題の対応を行う際にこの情報を役立てることができます。

特に以下の情報を記録することが必要です。

[a] 問題記述

-問題の症状

-受け取ったメッセージの完全な記述

-受け取ったSQLSTATE。SQLSTATEについては本ガイドの 2 章 2-2 節エラーコードが戻る場合、および『メッセージ解説書』を参照してください。

[b] 問題発生時に行っていた操作(これらも原因究明の手がかりとなります)

-問題を誘発したアクションおよびその時刻

-問題時に実行していたアプリケーション

-問題時に発行していた SQL ステートメントとその種類(静的/動的など)

[c] 問題発生前の作業

-パラメーターの変更

-ジョブの運用変更

-ピーク性の特徴(件数の増加、ユーザ数の増加)

-クライアントの追加

[d] ハードウェアのタイプ(マシンタイプ/シリアルナンバ)と物理構成/論理構成

[e] ソフトウェアのタイプとサービスレベルおよび適用した修正パック

[f] 問題発生後行った修正処置の結果

[g] サポートセンターに連絡された場合、その問題に割り当てられた番号

注:とくにアクションの時刻は、障害の資料の分析をする上で、非常に重要です。コマンド自体に、時刻を取得する機能があるもの (vmtune –t オプション、iostat –T オプション)を活用したり、また、機能が無い場合には、

```
$ /usr/bin/echo "# ¥c"; date +"%Y/%m/%d %H:%M:%S" ; <command>
```

あるいは、10 秒おきに実行する場合には、

```
$ while true; do /usr/bin/echo "# ¥c"; date +"%Y/%m/%d %H:%M:%S"; <command>; sleep 10; done
```

のようにして、必ず時刻を取得するようにしてください。

また、取得するファイル名に日時を含むようなことも検討してください。

上記の例では、日時のデータの出力フォーマットを、YYYY/MM/DD HH:MM:SS という形にしています。資料の見易さのために、時刻フォーマットをそろえることも検討してください。

2 問題の原因と症状の判別

2-1 はじめに

問題を解決するためには、その問題個所を判別することからはじめます。たとえば、問題は以下のいずれかにある場合があります。

- ハードウェア
- オペレーティング・システム
- ネットワーキング・システムまたは他のサブシステム
- DB2 サーバー
- DB2 クライアント
- DB2 以外のアプリケーション

2-1-1 最初に重点的にチェックする項目の選択

2-1-1-1 障害のきっかけがある程度判明している場合

発生した障害のきっかけとなった処理内容に最も関連していると思われる構成要素（OS、DB2、アプリケーション）について、最初に重点的にチェックを行なってください。

例えば、DB2のコマンドを実行した結果、何らかの障害現象が発生した場合は、まずはDB2に関する各チェック項目を重点的に調査して、何らかの想定ケースにその現象が合致する事が判明したら、その対応を行ないます。また、アプリケーションを実行した結果何らかの障害現象が発生したような場合は、まずはアプリケーションに関するチェックを重点的に始めてみてください。

もちろん、ある程度原因箇所に見当が付く兆候（エラーメッセージなど）が出ている場合は、その内容に従って、問題判別の作業を進めてください。(3 節エラーコードが表示される場合参照)

2-1-1-2 障害のきっかけが不明な場合

もし、その障害発生のきっかけとなる処理に、まったく見当が付かない場合は、その障害現象を発見した際の操作や状況を元に、調査を進めてみてください。

例えば、ログを“たまたま”見た時に、何らかのエラーメッセージを発見したが、そのエラーが発生した時に何の処理が実行されたのか全く判らないような場合は、そのエラーメッセージの内容や時間から推測して、OS、DB2、及び、アプリケーションの各構成要素のうち最も関連性の高い箇所から調べます。もし調査の手掛かりになる有用な情報がメッセージに含まれていない場合は、各構成要素のうちいずれかを適当に選択して調査し、判別が完了しなければ他の要素についても順次調査していきます。(2-3 節エラーコードは戻らないが、処理のレスポンスがない場合参照)

2-1-2 次にチェックする項目の選択

障害の特定に至らなた場合は、別の想定ケースを疑い、そのケースに合致するか否かのチェックを実施する手順に移ります。基本的には、障害現象を特定できるまで、この判別作業を繰り返します。

2-1-3 並行して問題判別が可能な場合

問題判別が実施可能な担当者が複数いる場合、例えば DB2 に関する調査とアプリケーションに関する調査を両者で並行して実施することによって、問題箇所の絞込み作業をより効率的に行なえることが期待できます。

2-1-4 問題判別を行う際の注意点

本書で示しているチェック項目や情報収集などを実行するコマンドの中には、メモリや CPU のリソースを大量に消費してシステムに重大なインパクトを与えてしまう可能性のあるもの(db2trc etc.)や、システムの状態を変えてしまうもの（db2untag, reorg etc.）があります。これらのコマンドを不用意に実行すると、障害の状況を悪化させてしまったり、状態が変わることによって当初発生した問題の判別が不可能になってしまう事も有り得ます。

よって、次に示すような影響度に応じて、各コマンドをレベル分けし、問題判別時に実行する順序を考慮する上での基準とすることとします。

（1） レベル 1

実行しても特にシステムに影響を与える可能性が殆ど無く、また、システムの状態に変更を加えることもないコマンド（またはチェック項目）

（2） レベル2

基本的には、実行しても特にシステムに重大な影響を与える可能性は少なく、システム状態に変更を加える可能性も殆どないが、何らかの条件と重なると影響を与える可能性が多少あるコマンド（またはチェック項目）

（3） レベル3

実行すると、CPU やメモリリソースを大量に消費するなどシステムに重大な影響を与えるか、システムの状態に変更を加えるコマンド（またはチェック項目）

＜各問題判別ツールの影響度に基づいたレベル分け一覧＞

	レベル1	レベル2	レベル3
OS	1) OS ﾛｸﾞﾁｪｯｸ(errpt) 2) ｼｽﾃﾑﾛｸﾞﾁｪｯｸ(syslog) 3) ﾌﾟﾛｾｽ稼働ﾁｪｯｸ(ps) 4) CPU/仮想ﾒﾓﾘﾁｪｯｸ(vmstat, sar) 5) ディスク I/O チェック(iostat) 6) N/W 状態ﾁｪｯｸ(netstat) 7) H/W ﾁｪｯｸ(errpt 等) 8) ﾃﾞｨｽｸﾁｪｯｸ(df, lslv,lspv, lsvg 等) 9) ﾌﾟﾛｾｽ間通信ﾁｪｯｸ(ipcs) 10) topas / nmon 11) truss	1) ﾌﾟﾛｾｽｱﾀｯﾁ(dbx) 2) svmon 3) vmtune 4) filemon / perfpmr	5) ﾌﾟﾛｾｽへのｼｸﾞﾅﾙ送信(kill) 6) ﾌﾟﾛｾｽ間通信の削除(ipcrm)
DB2	1) db2diag.log ﾁｪｯｸ(more, tail 等) 2) db2alert.log ﾁｪｯｸ(more, tail 等) 3) ｺｱ/ﾄﾗｯﾌﾟ・ﾌｧｲﾙ/ﾀﾞﾝﾌﾟﾌｧｲﾙの存在ﾁｪｯｸ(ls, more 等) 4) db2sysc 稼働ﾁｪｯｸ(ps) 5) GET DATABASE MANAGER CONFIGURATION 6) GET DATABASE CONFIGURATIOIN 7) db2expln/dynexpln 8) db2bfd 9) db2flsn 10) db2level 11) LIST TABLESPACES 12) LIST APPLICATIONS 13) db2mtrk 14) db2gcf –s (インスタンスの状態確認)	1) スナップショット・モニター 2) db2support 3) db2look 4) db2dart / inspect	1) DB2 トレース 2) DB2 コネクトトレース/DRDA トレース 3) CLI/ODBC/JDBC ﾄﾚｰｽ 4) ｲﾍﾞﾝﾄ・モニター 5) REORG/REORGCHK/RUNSTATS 6) BIND/REBIIND 7) RESTART DATABASE 8) db2untag 9) FORCE APPLICATIONS 10) db2_kill 11) ipclean 12) RESTORE/ROLLFORWARD 13) db2gcf (インスタンスの起動・停止)
AP	1) AP ﾛｸﾞﾁｪｯｸ(more, tail 等) 2) AP 稼働状態ﾁｪｯｸ(ps 等)	ﾌﾟﾛｾｽｱﾀｯﾁ(dbx)	1) AP ﾚﾍﾞﾙﾄﾚｰｽ(機能があれば) 2) ﾌﾟﾛｾｽへのｼｸﾞﾅﾙ送信(kill)

2-2 メッセージあるいは SQL エラーコードが表示される場合

2-2-1 一般的な対応方法

①戻されたエラーコードについて、以下の操作を実行しエラーの解説を参照して下さい。

```
$ db2 “? SQLxxxxx” | more          (← シェルのコマンドラインで実行する例)
$ db2 “? ASNxxxxx” | more          (← シェルのコマンドラインで実行する例)
db2 => ? CLIxxxxx                  (← ＣＬＰプロンプトで実行する例)
```

<SQL メッセージ表示例>

```
db2 => ? SQL0968C
SQL0968C ファイル・システムがいっぱいです。
説明: データベースを持っているファイル・システムのいずれかがいっぱいです。このファイル・
システムには、データベース・ディレクトリ、データベース・ログ・ファイル、または表スペース・
コンテナーが含まれている可能性があります。
ステートメントは処理されません。
ユーザーの処置: 診断ログを参照して、いっぱいになったファイル・システムを判別してください。
～～～～　途中、省略　～～～～

sqlcode: -968
sqlstate: 57011
```

②SQLSTATE コードがある場合、更に以下の操作を実行して、解説を参照してください。

```
$ db2 “? NNNNN” | more    (← シェルのコマンドラインで実行する例)
db2 => ? NNNNNN           (← ＣＬＰプロンプトで実行する例)
```

<SQLSTATE メッセージ表示例>

```
db2 => ? 57011
SQLSTATE 57011: 仮想記憶またはデータベース資源が使用できません。
```

③ 『メッセージ解説書』にもエラーコード別の解説が存在します。

④ よく発生するエラーの対応方法については 3 節エラーコードが表示される場合の対応方法を参照してください。

2-3 メッセージは表示されないが、処理のレスポンスがない場合の問題判別

2-3-1 はじめに

2-3-1-1 レスポンスのない状態の分類

エラーは特に返されないが何の応答もない場合に考えられる状態を、ここでは大きく次のように分類して取り扱うこととします。

- ダウン

(A)インスタンスダウン(5-1 節インスタンスダウンの可能性が高い場合の対応方法参照)

DB2インスタンスのプロセス(またはスレッド)本体が停止している状態

(B)アプリケーションダウン(6-1 節アプリケーションに問題がある可能性が高い場合の対応方法参照)

アプリケーションのプロセス(またはスレッド)本体が停止している状態

- 処理中(5-2 節処理中の可能性が高い場合の対応方法参照)

(C)処理遅延

処理は進行中であるが、通常時よりも非常に進み方が遅い状態(CPU は使用している)

(D)ループ

同じ処理を延々と繰り返す状態で、そのまま放置していても、その状態から脱しない状態
(CPU は使用している)

- 非処理中

(E)ウェイト(5-3 節ウェイトの可能性が高い場合の対応方法参照)

何らかの処理がされるのを待機していて、それ自身の処理は中断している状態
(CPU は基本的には使用していない)

一般的には、例えば次のような状態が考えられます。

1) OS レベルリソース待ち(CPU ディスパッチ、セマフォ、プリンタスプーリング等)
2) アプリケーションレベルのリソース待ち(ロック等)
3) 割込み待ち(アラーム、デバイス割込み等)
4) ユーザーアクション待ち(キーボード、マウス入力など)

DB2に関しては、表データのロック待ち(5-3-1 節参照)や、ログフル時のブロッキング設定(5-3-2 節参照)、及び、write suspend 時の物理ディスク書込み発生時のブロッキング(5-3-2 節参照)などの状態が考えられます。

(F)ハング(5-4 節ハングの可能性が高い場合の対応方法参照)

上記のいずれにも当てはまらない状態。(CPU は基本的には使用していない)

※ある処理が停止しているとき、実際には別の何らかの処理を「ウェイト」している可能性もあるが、何を「ウェイト」しているのか明確に判別できない場合は、「ハング」と見なす立場をとります。

2-3-1-2 各状態の判別に関する考え方

各状態を判別する為の手法は、基本的には、アプリケーションの稼働状態やリソースの負荷状態を見るほか、実行中のトレースやスナップショットなどのデータをサンプリングし、その結果を総合的に判断することになります。

ここで注意すべき点は、状況データをサンプリングする回数とタイミング（インターバルの取り方等）によっては、「正しい」判断をする為に必要なデータが得られない可能性がある点です。
しかし、どのようなシステムでも適用できる“適切なサンプリング回数とタイミング”を一般的に決めることは不可能である為、本資料の判別手順に記載している例は、あくまでも、ひとつの目安として捉えていただく必要があります。
実際にユーザーのシステムにおいて当資料の判別方法を適用する際には、状況データのサンプリング回数およびタイミングは、ユーザーの判断に委ねることが前提条件となりますことをご了承ください。
また、取得した状況データに対して、例えばその結果が「通常よりも遅い」と判断する基準などについても、あくまでもユーザーの判断に委ねることになります。
以下の節以降で、具体的な問題判別手順を記載します。

2-3-2 OS レベルの確認

障害の状況を判別するために OS レベルでは以下の項目を順番に確認してください。

2-3-2-1 OS エラーログ確認

```
$ errpt  | more
```

ある程度見当が付くエラーが出ている場合、以下の例を参考にそのエラーについて調査してください。

[a] H/W の障害

“T(エラータイプ)”の列に"P(Permanent or Performance or Pending)"、かつ“C(エラークラス)”の列に"H(Hardware)"の文字が表示された場合、H/W障害の可能性が高いです。4-1 節HWに問題がある可能性が高い場合を参照してください。

[b] ファイルシステムFULLの表示がある場合、ファイルシステムの空き容量を確認するため 2-3-2-5 節ファイルシステム空き容量確認を参照してください。

[c] ソフトウェアプログラムが異常終了の表示がある場合、まず errpt -a を確認してください。

その際、プログラム名の欄にDB2 関連の表示があった場合、DB2 に問題のある可能性が高いです。
2-3-3 節DB2 レベルの確認を参照してください。
その他の表示がされる場合、内容により見当をつけ該当のページを参照してください。

2-3-2-2 CPU/仮想メモリ確認

```
$ vmstat 2 30
```

```
kthr     メモリ              ページ                フォルト        CPU
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0 210418 69108   0   0   0   0    0   0  61   86   3  0  1 97  2
 0  1 210418 69108   0   0   0   0    0   0 953  683 995  0  0 99  0
 0  1 210418 69108   0   0   0   0    0   0 960  588 995  0  1 99  0
```

【解説】

- vmstat が出力する統計データの最初の行は、OS 起動時から現在までの平均値であり殆ど参考にならない為、現時点の負荷状況としては2行目以降を参照してください。
- vmstat が出力する統計データでは、システム全体の平均が表示される為、マルチプロセッサシ

ステム(SMP)の個々のプロセッサ毎の負荷を調べることはできません。SMP のプロセッサ毎の活動を調べたい場合は、sar コマンド(sar -P ALL 等)を使用してください。

出力状況別に以下のような対応を行ってください。

[a] CPU使用率(us+sys)が 0%を維持している場合

CPUは使用されていないが、Disk I/Oは行われているかの確認を行います。2-3-2-4 節Disk I/O確認を参照してください。

[b] CPU使用率(us+sys)が 100%近い状態を継続している場合

db2 プロセスに原因の可能性がある場合、DB2DIAG.LOGの確認を行います。2-3-3-2 節DB2DIAG.LOGの確認を参照してください。

[c] CPUのI／O待ち率(vmstat の wa)が通常の平均時よりも高い状態（例えば 50%以上）が継続している場合

Disk I/Oの状況を調査します。2-3-2-4 節Disk I/O確認を参照してください。

[d] ページング(vmstat の pi と po)が頻発(例えば 5 以上)している場合

プロセスごとのページングを確認してください。

```
$ ps avxw
```

```
PID    TTY STAT   TIME PGIN  SIZE   RSS   LIM  TSIZ   TRS %CPU %MEM COMMAND
  0      - A   107:27    7   276   208    xx     0     0  0.0  0.0 swapper
  1      - A   10:18    79   620   636 32768    25    36  0.0  0.0 /etc/init
516      - A   172039:49 0   272   192    xx     0     0 24.6  0.0 kproc
774      - A   172539:48 0   272   204    xx     0     0 24.7  0.0 kproc
1032     - A   173248:22 0   272   204    xx     0     0 24.8  0.0 kproc
1290     - A   173599:31 0   272   212    xx     0     0 24.9  0.0 kproc
1548     - A    1:45     0   276   188    xx     0     0  0.0  0.0 kproc
1806     - A    4:16     0   280   232    xx     0     0  0.0  0.0 kproc
2064     - A   102:55    0   328   272    xx     0     0  0.0  0.0 kproc
2904     - A    0:00     0   280   232    xx     0     0  0.0  0.0 kproc
3086     - A    0:00    55   964  1044 32768    36    56  0.0  0.0 /usr/sbin/portmap
```

プロセス毎にページングの回数（PGIN:プロセス起動後の合計）を確認後、異常なプロセスがあればその原因を追求してください。

db2 プロセスが異常な場合、db2diag.logを確認します。2-3-3-2 節DB2DIAG.LOGの確認を参照してください。

2-3-2-3 N/W 状態確認

```
$ netstat -i
```

```
Name  Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
lo0   16896 link#1                        1255390     0  1255395     0     0
lo0   16896 127         localhost          1255390     0  1255395     0     0
lo0   16896 ::1                            1255390     0  1255395     0     0
en0   1500  link#2      0.4.ac.49.2a.b8          0     0   349188     0     0
en0   1500  192.168.1   yagi08                   0     0   349188     0     0
tr0   1492  link#3      0.4.ac.61.d0.b8   30957628     0 15007990    92     0
tr0   1492  9.116.235   yagi08.hakozaki.i 30957628     0 15007990    92     0
et0*  1492  link#4      0.4.ac.49.2a.b8          0     0        0     0     0
```

[a] ネットワークインタフェースは全て使用可能であるかを確認してください。

インタフェースに*印がついているときはそれは使用不可能状態を表します。

[b] サーバー側のネットワークトラフィックが上がっていないかを確認してください。

※ネットワーク経由でアクセスしたときにハングしたように見えるのであれば、この項目をチェックしてください。

【解説】

Oerrs 欄が Opkts の 1%より大きい時はインターフェースの負荷が大きすぎます。

Oerrs 欄が lpkts の 1%より大きい時は netstat –m によりメモリ不足がないかのチェックを行ってください。

2-3-2-4 Disk I/O 確認

```
$ iostat -d  2 30
```

```
Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.0       0.0       0.0          0         0
hdisk1           0.0       0.0       0.0          0         0
hdisk2           0.0       0.0       0.0          0         0
hdisk3           0.0       0.0       0.0          0         0
cd0              0.0       0.0       0.0          0         0
```

出力状況別に以下のような対応を行ってください。

[a] ディスク I/O が停止(全ての値が 0)している場合

OSのハングの可能性が高いです。4-2 節OSに問題がある可能性が高い場合を参照してください。

[b] ディスク I/O が多発している場合

I/O が多発しているディスクの用途を確認してください(lspv、lsvg、df、lsps –a：ページングスペースの物理ディスク)

- DB2 が使用しているディスクの使用率が高い場合

  DB2 に問題がある可能性が高いです。2-3-3 節DB2 レベルの確認を参照してください。

- DB2 以外が使用しているディスクの使用率が高い場合

  他のアプリケーションに問題がある可能性が高いです。2-3-4 節アプリケーションレベルの確認を参照してください。

2-3-2-5 ファイルシステム空き容量確認

```
$ df -k| more
```

```
Filesystem    1024-blocks      Free ***%Used***    Iused %Iused Mounted on
/dev/hd4            81920     43360   48%     2533    13% /
/dev/hd2          6586368    101880   99%    66165     9% /usr
/dev/hd9var        163840       152  100%      761     4% /var
/dev/hd3            65536     21360   68%      473     6% /tmp
/dev/hd1            98304     90064    9%      182     2% /home
/dev/sw_users     1015808    326008   68%     4430     4% /sw_home
/dev/lvdatabase   4194304    614048   86%    13693     3% /database
```

%Used の数値をもとにファイルシステムの空き容量を確認してください。

[a] DB2DIAG.LOG 用のファイルシステムに不足がある場合

3-1-1-4 節SQL0964C(データベースのトランザクション・ログがいっぱいを参照してください。

[b] DB2 データ用のファイルシステムに不足がある場合

3-1-1-5 節SQL0968C(ファイルシステムがいっぱい)を参照してください。

[c] 上記以外のファイルシステムに不足がある場合

4-3 節空き領域不足の場合を参照してください。

2-3-3 DB2 レベルの確認

障害の状況を判別するために DB2 システムレベルでは以下の項目を順番に確認してください。

2-3-3-1 DB2 プロセスの稼働の確認

ps コマンドにより db2sysc プロセスの稼働状況を監視してください。

$ ps -ef | grep <*instance-name*>

※正常な場合の画面出力例

```
<$db2instance>  48670 100444   3 15:15:19  pts/2  0:00 grep <$db2instance>
<$db2instance>  66766  91032   0 12:26:42      -  0:00 db2gds 0
<$db2instance>  72910 100444  16 15:15:19  pts/2  0:00 ps -ef
<$db2instance>  84124  66766   0 12:26:43      -  0:00 db2srvlst 0
<$db2instance>  91032 142110   0 12:26:42      -  0:00 db2sysc 0
<$db2instance>  93168  66766   0 12:26:43      -  0:00 db2resyn 0
<$db2instance>  96070 136316   0 12:26:47      -  0:00 db2agent (instance) 0
<$db2instance> 100076      1   0 12:26:27      -  0:00 /<$INSTHOME>/sqllib/bin/db2bp 100444 5
<$db2instance> 100444 126906   1 12:26:09  pts/2  0:00 -ksh
<$db2instance> 136316  91032   0 12:26:42      -  0:00 db2ipccm 0
```

[a] db2sysc が存在しない場合

DB2 インスタンスダウンの可能性が高いです。5-1 節インスタンスダウンの可能性が高い場合の対応方法を参照してください。

2-3-3-2 DB2DIAG.LOG の確認

①データベースマネージャー構成パラメーターの DIAGPATH を確認します。

(デフォルトは$ISNTHOME/sqllib/db2dump)

$ db2 get dbm cfg | grep DIAGPATH

診断データのディレクトリー・パス (DIAGPATH) = <*diag_path*>

②db2diag.log の内容を確認します。

$ view <*diag_path*>/db2diag.log

[a] エラー表示があった場合、内容により見当をつけ該当のページを参照してください。

2-3-3-3 システムコアファイルの存在確認と解析

システムコアファイルの説明は 1 章 2-3-3 節coreを参照ください。

【確認手順】

以下の手順でシステムコアファイルの検索とその作成日付の確認します。システムコアファイルの名前は"core"で、アプリケーションが実行されているディレクトリーに出力されます。

① システムコアファイルの検索とその作成日の確認

find <*path*> -name core - print

cd <*core_path*>

ls -l core

※ファイルは必ず保管してください。

② core ダンプが発生するとエラーログに CORE_DUMP のエラーラベル名で記録されるので、その記録より実行プログラム名は判別可能です。

③ dbx コマンドを使用して、どの関数がコアファイルの作成の原因となったかを判別できます。これは、データベースマネージャーまたは DB2 コネクトがエラーを起こしたかどうかを識別したり、問題の原因はオペレーティング・システムまたはアプリケーションのエラーなのかどうかを識別するのに役立つ簡単な検査です。

コアファイル・ダンプが起きる原因となった関数を判別するためには、次のコマンドを入力してください。

```
$ dbx <program_name> <core_filename>
```

ここで、<*program_name*>は、異常終了したプログラムの名前であり、<*core_filename*>は、 コアファイル・ダンプを含んでいるファイルの名前です。 <*core_filename*>パラメーターはオプションです。 これを指定しなければ、デフォルト名の"core" が使用されます。 dbx は、core ファイルによって該当プログラムのコンパイル時の -g オプションの有無を認識しており、 -g なしでコンパイルされている場合は、dbx コマンドの機能が一部制限されます。dbx コマンドを終了させるためには、dbx プロンプトに quit を入力します。 dbx コマンドには、 この節で説明されているよりもさらに多くの関数を提供します。 それらを知るには、OS にコマンドマニュアルがインストールされている場合には、UNIX ベースのコマンド・プロンプトから man dbx を入力してください。

【解析手順例】

次の例は、dbx コマンドを使用して、 "main" という名前のプログラムについてのコアファイルを読み取る方法を示しています。

① コマンド・プロンプトから、次のように入力します。

```
$ dbx main
```

② 以下と似た出力が、画面に表示されます。

```
dbx version 3.1 for AIX.
Type 'help' for help.
   reading symbolic information ...
 [using memory image in core]
segmentation.violation in freeSegments at line 136
136 (void) shmdt((void *) pcAdress[i]);
```

上記の例では、メモリー・ダンプを引き起こした関数の名前は "freeSegments" です。 関数名が "db2"、"sql"、または "ddcs" で始まっている場合、 エラーがデータベースマネージャーまたは DB2 コネクト製品で起きた可能性があることを示します。

③ 障害個所へのプログラム・パスを表示するために、dbx プロンプトから where を入力します。

```
(dbx) where
```

```
freeSegments(numSegs = 2, iSetId = 0x2ff7f730, pcAddress = 0x2ff7f758, line136
in "main.c"
main (0x1, 2ff7f7d4), line 96 in "main.c"
```

この例では、"main.c" の第 96 行から呼び出された freeSegments の第 136 行でエラーが起

きました。

④ dbx コマンドを終了させるためには、dbx プロンプトに quit を入力します。

(dbx) quit

2-3-3-4 ダンプ・ファイル/DB2 コアファイルの存在確認と解析

ダンプ・ファイル/DB2 コアファイルの詳細は 1 章 2-1-3 節DB2ダンプ・ファイル/DB2 コアファイルを参照ください。

【確認手順】

① db2diag.log によりダンプ・ファイルの出力確認

$ view <*diag_path*>/*db2diag.log*

ダンプ・ファイル出力表示があれば、該当パスへ

<db2diag.log の出力例>

この例では、/home/db2inst1/sqllib/db2dump/118328.000 という名前のファイルがダンプ・ファイルに該当します。

```
------------------------------------------------------------------------------------------------
----------------------
2002-06-13-21.08.48.300950   Instance:db2inst1   Node:000
PID:118328(db2agent (TESTDB))   Appid:*LOCAL.db2inst1.020613120344
relation_data_serv  sqlrr_signal_handler   Probe:10   Database:TESTDB
DIA7107I Execution of a component signal handling function has begun.
Dump File:/home/db2inst1/sqllib/db2dump/118328.000 Data:Recursive dump_sect   【１】
------------------------------------------------------------------------------------------------
----------------------
```

② DB2 コアファイルの検索とその作成日付の確認

※DB2 コアファイルはデータベースマネージャー構成パラメーターの DIAGPATH に作成されることが多いです。例えば、<*diag_path* >/c63422.000 というディレクトリーが存在した場合、pid 63422 のプロセスのコアファイルを含むディレクトリーです。

$ find $INSTHOME/. -name c*.000 -print

$ ls-l < *diag_path* >/c*

[a] 該当ファイルが存在した場合

DB2 に問題がある可能性が高い場合、下記手順も順に実行してください。

※ファイルは必ず保管してください。

【解析手順】

DB2 コアファイルはシステムコアファイルと同様に解析可能です。手順は 2-3-3 節システムコアファイルの存在確認と解析を参照してください。

2-3-3-5 DB2 トラップ・ファイルの存在確認

DB2 トラップ・ファイルの詳細は 1 章 2-1-4 節DB2 トラップ・ファイルを参照してください。

【確認手順】

①データベースマネージャー構成パラメーターの DIAGPATH を確認します。

※DB2トラップ・ファイルはデータベースマネージャー構成パラメーターの DIAGPATH で指定されたパスに出力されます。

```
db2 get dbm cfg | grep DIAGPATH
   診断データのディレクトリー・パス               (DIAGPATH) = < diag_path >
```

②DB2トラップ・ファイルの存在とその作成日付を確認します。

※トラップ・ファイルの最初の文字は"t"で、その後にプロセス ID(pid)が続きます。例えば、t62122.000 は pid62122 のプロセスのファイルです。

```
$ cd  <diag_path>
$ ls -l
```

[a] 該当ファイルが存在した場合

DB2 に問題がある可能性が高い→下記手順も順に実行

※ファイルは必ず保管してください。

2-3-3-6 DB2 基本的なコマンドの実行

以下の 4 つのコマンドが実行できるかを確認

```
$ db2 list applications
$ db2 get dbm cfg
$ db2 get db cfg for <database-alias>
$ db2 connect to <database-alias>
```

[a] 全て実行可能な場合

インスタンス及びデータベース全体のレベルでは特に問題無しと予測されます。

[b] エラーが発生した場合

メッセージ内容から見当をつけて、対応を行なってください。

2-3-3-7 DB2 アプリケーションの活動状況のリスト

以下のコマンドを実行してください。

```
$ db2 list applications for <database-alias> show detail
   TESTDB mainD.exe  498  AC12144B.0AA0.000510221225 0001 1 0 62034 UOW Executing
    収集されませんでした TESTDB  $INSTHOME/NODE0000/SQL00001/
```

【解説】

Applicatoin status の表示により、実行中(UOW-Executing)か、ロック待機(Lock-Wait)か、ユーザー入力待機(UOW-Waiting)かの判断可能

[a] Application status ：Lock-Wait のプロセスがある場合

ロック待ちの可能性があります。5-3-1 節ロック待ち(Lock-Wait)のプロセスが発生して処理が進まない場合を参照して下さい。

2-3-3-8 DB2 スナップショットの収集

以下の手順でスナップショットを取得してください。

①間隔を空けて数回、スナップショットを取得してください。

```
$ db2 get snapshot for all applications > snapshot_app1.log
$ sleep 30
$ db2 get snapshot for all applications > snapshot_app2.log
$ sleep 30
$ db2 get snapshot for all applications > snapshot_app3.log
```

②ログの確認を行ってください。

```
$ view snapshot_app*.log
```

[a] エラーが発生した場合、エラー内容から見当をつけて、対応を行ってください。

[b] Application status ：Lock-Wait のプロセスがある場合

```
           Application Snapshot
Application handle                         = 49
Application status                         = Lock-Wait
Status change time                         =
Application code page                      = 819
```

ロック待ちの可能性があります。5-3-1 節ロック待ち(Lock-Wait)のプロセスが発生して処理が進まない

```
           Application Snapshot
Application handle                         = 49
Application status                         = Lock-Wait
Status change time                         =
Application code page                      = 819
```

```
           Application Snapshot
Application handle                         = 49
Application status                         = Lock-Wait
Status change time                         =
Application code page                      = 819
```

```
           Application Snapshot
Application handle                         = 49
Application status                         = UOW-Executing
Status change time                         =
Application code page                      = 819
                    :
                    :
Rows deleted                               = 0
Rows inserted                              = 0
Rows updated                               = 0
Rows selected                              = 0
Rows read                                  = 0
Rows written                               = 0
```

場合を参照して下さい。

[c] Application status ：UOW –Executing のプロセスがあり、かつ Rows-Read/Written に変化がある場合

処理中の可能性が高いです。5-2 節処理中の可能性が高い場合の対応方法を参照してください。

[d] Application status ：Rollback Active のプロセスがありかつ Rows-Read/Written に変化がある場合

以下の可能性があります。

- DB2_BLOCK_ON_LOG_DISK_FULL=ON 設定時のログディスクフルによる待機の可能性

  5-3-2 節を参照してください。
- set write suspend 時のディスク書込み要求発生による待機の状態の可能性

5-3-2 節を参照してください。

[e] Application status ：UOW-Waiting のプロセスがある場合

アプリケーションプロセスの状態をチェックします。アプリケーションが”処理中”か”停止中”かをチェックしてください。

① アプリケーションの名称(実行ユーザーの名称)からプロセス ID を確認してください。

```
$ ps -ef | grep <app_name>
```

② 以下のコマンドによりトラップ・ファイルが、データベースマネージャー構成パラメーターの DIAGPATH 以下に出力されます。

```
$ kill -36 <app_pid>
$ mv <trapfile> <backup-trapfile>
$ sleep 30
$ kill -36 <app_pid>
$ mv <trapfile> <backup-trapfile>
$ sleep 30
$ kill -36 <app_pid>
```

diff コマンドなどで、出力されたトラップ・ファイルに変化があるかの解析を行ってください。

③ トラップ・ファイルに変化があれば”処理中”と判断し、変化無しならば”停止中”と判断してください。

- 処理中ならば、「処理遅延」「ループ」、それとも「正常に進行中」なのかのいずれか。 アプリケーションに依存するが、進行中であることを示すもの（例えばアプリケーションログ）があればそれでチェックする。もしなければ、5-2 節処理中の可能性が高い場合の対応方法を参照してください。
- 停止中ならば、何かを「ウェイト」しているのか「ハング」しているのかのどちらかであるため、まず 5-3 節ウェイトの可能性が高い場合の対応方法を参照し、該当しなければ 5-4 節ハングの可能性が高い場合の対応方法を参照してください。

※前記のスクリプトで採取したスナップショット結果のファイルを、diff コマンドで比較し、変化したモニタリング項目を表示することも可能です。

```
$ diff snapshot_app1.log    snapshot_app2.log
```

```
< Application status                        = UOW Waiting
< Status change time                        = 06/14/2002 07:29:22.191710
---
> Application status                        = UOW Executing
> Status change time                        = 06/14/2002 07:29:34.58390721c21
< Application idle time                     = 2 seconds
---
> Application idle time                     =37c37
< Snapshot timestamp                        = 06/14/2002 07:29:24.673523
---
> Snapshot timestamp                        = 06/14/2002 07:29:34.929217
～～～～～  途中、省略  ～～～～～

< Rows read                                 = 4
---
> Rows read                                 = 31270
～～～～～  途中、省略  ～～～～～
```

2-3-4 アプリケーションレベルの確認

障害の状況を判別するためにアプリケーションレベルでは以下の項目を順番に確認してください。その際以下が前提条件となります。

- アプリケーションのプロセス名称があらかじめわかっていること
- アプリケーション側のログの出力場所

※以上の項目で可能なものは日常の運用で確認をしておく必要があります。

2-3-4-1 アプリケーションプロセスの稼働確認

ps コマンドを用い、アプリケーションのプロセス名称をもとに稼働状況を監視してください。

$ ps –ef | grep <*app_name*>

[a] アプリケーションが存在しない場合

アプリケーションダウンの可能性が高いです。6-1 節アプリケーションに問題がある可能性が高い場合を参照してください。

2-3-4-2 アプリケーション側のログの確認

[a] ログにより判断可能な場合

適切な処理を実行してください。

3 エラーコードが表示される場合の対応方法

以下に、発生しやすくかつ得られたメッセージのみでは対応の困難な場合のあるエラーとその対応方法を記載します。

3-1-1-1 SQL0294N(コンテナーはすでに使用中)

【解説】

表スペースコンテナーを作成しようとした際、DB2 はすでに他の表スペースで使用中のコンテナーを誤って使わないようタグで制御しています。

他の表スペースでは使用していないことが確実であるのにこのエラーが出る場合は、以前表スペースコンテナーとして使用していたが、何かしらの理由で表スペースの削除、あるいはその際のタグの除去が成功していなかった可能性があります。

【注意事項】

下記対応手順は他の表スペースで使用中でないことが確実である場合のみ実施してください。使用中だった場合は、使用しているデータベース、及び、新しく使用しようとしたデータベースの両方が損傷を受けますので、ご注意ください。

【対応方法】

① 論理ボリュームをオープン(使用)しているプロセスが存在するかの確認

・オープンしているプロセスの特定をして下さい。

```
$ fuser -u /dev/<vname>
```

・存在していればプロセスを停止して下さい。

```
$ kill -9 <pid>
```

②タグの削除

```
$ db2untag  /dev/<vname>
```

<出力結果>

```
db2untag: A service tool to remove the DB2 tag on a tablespace container.

The tag is used to prevent DB2 from reusing a container
in more than one tablespace.
If a tablespace/database is destroyed thru unnatural means, then the tag can be
left behind preventing future DB2 use of the resource.

WARNING: This tool should only be used by informed sysadmins.

Using file </dev/vname>

version  = 213
db seed  = 6541E162
poolID   = 2
contID   = 0
created  = 0
used     = 1
poolLSN  = 0000 0000 0684

Instance = inst
```

```
Database = SAMPLE

If you are sure that this container is no longer needed by the
identified DB2 database, then answer 'Yes'.
Do you want to untag the container /dev/lvname?
---> Yes
Tag removed
```

3-1-1-2 SQL0902C(システム・エラーが発生し、後続の SQL ステートメントは処理されない)

以下にこのエラーを出力する原因となる障害例とその対応を記載いたします。

<想定例1> SYSCATSPACE(カタログ表スペース)に問題が発生した場合

【解説】

SYSCATSPACE が破損、消失した場合、データベース上で稼動しているアプリケーションに影響があります。こういった場合に、既に接続済みのアプリケーションが SELECT 文を実行するとこのエラーが表示されます。

新たにそのデータベースに接続するアプリケーションには「SQL0293N 表スペース・コンテナーのアクセス・エラーです。」のエラーが返されます。

【対応方法】

SYSCATSPACE の回復手順は以下となります。

①db2 list history backup コマンドで最新のバックアップとログを確認

```
$ db2 list history backup all for <database-alias>
```

②①で確認したバックアップ・場所にバックアップ・ファイルがあることを確認

```
$ ls
SAMPLE.0.inst. NODE0000.CATN0000. 20020621181335.001
```

③①で確認した"Earliest Log"以降のログがログ・パスかアーカイブログ・パス(USER EXIT 使用)にあることを確認

```
$ db2 get database cfg for <database-alias> | grep パス
ログファイルのパス      = /database/NODE0000/SQL00001/SQLOGDIR/
$ ls / database/NODE0000/SQL00001/SQLOGDIR/
S0000005.LOG S0000006.LOG
```

④データベースへの接続を停止(SYSCATSPACEの回復はオフラインで実行しなければならない)

```
$ db2 force application all
```

⑤ロールフォワード回復を実行(1 章 1-4-4-2 節表スペース単位のロールフォワード回復参照)

※ SYSCATSPACE は最新時点(to end of logs)に回復させなければなりません。

3-1-1-3 SQL0911N(デッドロックまたはタイムアウトのため、現在のトランザクションが ロールバックされた)

【解説】

同じデータベースに接続した 2 つまたはそれ以上のアプリケーションがロック待ちしているときにタイムアウトまたはデータベースのデッドロックが発生したことが考えられます。メッセージの後半に出力される理由コードによりタイムアウト(理由コード 68)かデッドロック(理由コード 2)か判別できます。その時の作業単位は、ロールバックされています。よって、現在は処理が続行されているはずですが、その場合、再発防止策として以下の対応を行ってください。

また、現在処理のレスポンスがない場合は、ロック待ちによりアプリケーションが待機しつづけている可能性があるため、5-3-1 節ロック待ち(Lock-Wait)のプロセスが発生して処理が進まない場合を参照してください。

【対応方法】

<デッドロック発生のアプリケーション特定方法>

デッドロックを発生させたアプリケーションを特定するには、イベントモニターを使用します。

(デッドロックが発生する前にイベントモニターを起動しておく必要があります。DB2 V8 からは、デフォルトで、DB2DETAILDEADLOCK という名前のイベントモニターが起動しており、これを使用する場合、以下の⑤以降のみを実施することになります。)

① イベントログを保存するディレクトリーを作成します。

```
$ mkdir 〈log-directory〉
```

② イベントモニターを以下の手順で作成しておきます。

```
$ db2 connect to 〈database-alias〉
$ db2 “create event monitor 〈monitor-name〉
      for deadlocks,statements
      write to file '〈log-directory〉'
      maxfiles none maxfilesize 500
      replace “
```

モニターは一度作成しておけば、何回でも起動/停止が行えます。

③ 作成したイベントモニターを起動します

```
$ db2 connect to 〈database-alias〉
$ db2 set event monitor 〈monitor-name〉 state 1
```

この時、イベントモニターを起動したセッション(ターミナル画面)を終了してはいけません。

④ イベントモニターを起動した状態で、アプリケーションを実行し、デッドロックデッドロックが発生したとします。

⑤ イベントモニターを停止します

イベントモニターを起動したターミナル画面で、以下のコマンドを実行します。

```
$ db2 flush event monitor 〈monitor-name〉 buffer
$ db2 set event monitor 〈monitor-name〉 state 0
```

⑥ イベントログを整形します

```
$ db2evmon -db 〈database-alias〉 -evm 〈monitor-name〉 > 〈outputfile〉
```

もしくは

```
$ db2evmon -path 〈log-directory〉 > 〈outputfile〉
```

上記コマンドで作成される整形されたイベントログをチェックする事でデッドロックを起こしているアプリケーションを特定できます。

⑦ <*outputfile*> をエディタ等で開き、Deadlock Event ...から始まるセクションを探します

14) Deadlock Event ...

Number of applications deadlocked: 2

Deadlock detection time: 06/26/2002 14:57:24.983475

⑧ Deadlock Event セクションの下にある、Deadlocked Connection セクションの内容から、Deadlock を起こしたアプリケーションのアプリケーション ID を得る事ができます

15) Deadlocked Connection ...

Appl Id: *LOCAL.db2inst1.020626051940 (←アプリケーション ID)

⑨ アプリケーション ID をキーに<*outputfile*>内の Connection Header Event セクションを検索すると、アプリケーションの情報を得る事ができます。

4) Connection Header Event ...

Appl Handle: 17

Appl Id: *LOCAL.db2inst1.020626051940 ←アプリケーション ID

Appl Seq number: 0001

DRDA AS Correlation Token: *LOCAL.db2inst1.020626051940

Program Name : db2bp ←プロセス名

<デッドロックの防止策>

デッドロックを発生させたアプリケーションの特定後、以下の対応が考えられます。

① デッドロックに関係するアプリケーション同士で、オブジェクトへのアクセス順番を揃えるようにしてください。

② デッドロックを起こしやすいアプリケーションに対して、作業単位が終了しだい COMMIT を発行するように修正してください。

③ ロックリストのサイズが適正であるかの確認

ロックリストが不足していると、ロックエスカレーション(通常ロックは行単位だが、ロック用メモリーエリアが不足した場合、表単位のロックに自動的に変わること)が発生し、長時間のロック待ちやデッドロックを引き起こす可能性があります。

以下のステップは、データベース構成パラメーターの LOCKLIST に必要なページ数を決定するのに役立ちます。

1. ロック・リストのサイズの下限を計算します。ユーザー環境により、以下の計算式の中からひとつを使用して計算します:

(ア) (a * x * maxappls) / 4096

(イ) 集線装置(Concentrator)が使用可能になっている場合:

(a * x * max_coordagents) / 4096

(ウ) 集線装置が使用可能になっているパーティション・データベースの場合:

(a * x * max_coordagents * データベース・パーティションの数) / 4096

ただし、a はアプリケーション当たりの平均ロック数の見積もりであり、x はそれぞれのロックに必要なバイト数 (32 ビット・プラットフォームでは 40 バイト、64 ビット・プラットフォームでは 64 バイト) です。

2. ロック・リスト・サイズの上限を計算します。

   (a * y * maxappls) / 4096

   y はオブジェクトに対する最初のロックに必要なバイト数です。 (32 ビットのプラットフォームの場合は 80 バイトで、64 ビットのプラットフォームの場合は 128 バイトです。)

3. データに対する並列量を見積もり、また計算した上限と下限の間になるように、予測に基づいて locklist の初期値を選択します。
4. データベース・システム・モニターを使用してこのパラメーターの値を調整します。データベース・システム・モニターを使用すると、指定したトランザクションによって保持される最大ロック数を判別することができます。locks_held_top (保留されているロックの最大数) モニター・エレメントを参照してください。

ここで、MAXAPPLS は1データベースあたりの最大同時稼動アプリケーション数です。データベース構成パラメータに含まれます。

ロックリストの更新は以下のコマンドを実行してください。

$ db2 update db cfg for <*database-alias*> using LOCKLIST <*size*>

なお、ロックエスカレーションの発生回数の情報はデータベースのスナップショット・モニターで取得できます。

④ アプリケーションが-911 戻りコードを適切に処理できるようにコード化されているかを確認してください。

⑤ 分離レベルの指定

- SELECT ステートメントを発行するアプリケーションで分離レベル UR(非コミット読み取り)を指定する方法もあります。UPDATE ステートメントを発行するアプリケーションで排他ロックした場合でも問題なく、SELECT ができます。ただし、分離レベル UR では、COMMIT されていない行まで読み取ってしまいます。この動作がアプリケーションの動作に影響を与えない範囲で分離レベルを変更してください。
- 分離レベル RR がインスタンスで不要ならば DB2_RR_TO_RS 変数を YES に設定してください。RR 実現のためにかかるロックの負荷を軽減することができます。

  db2set DB2_RR_TO_RS=YES

  DB2 再起動

3-1-1-4 SQL0964C(データベースのトランザクション・ログがいっぱい)

【解説】

トランザクションがデータベース構成パラメーターの Logprimary+Logsecond 分のアクティブ・ログを使い切ってしまっている場合にこのエラーが表示されることが多いです。この場合、更新処理は ROLLBACK されることにより、アクティブ・ログがアーカイブ・ログとなり、新規のアクティブ・ログの割り振りが可能になります。再発防止のため、主に以下の対応を行います。

- アクティブ・ログファイル容量(データベース構成パラメーターの Logfilsz、Logprimary、Logsecond)の適正な値への増加
- アプリケーションによる COMMIT/ROLLBACK の増加

【対応方法】

① db2diag.log より確認を行います。

```
$ db2 get dbm cfg | grep DIAGPATH
   診断データのディレクトリー・パス               (DIAGPATH) = <diag_path>
$ view <diag_path>/db2diag.log
```

以下のエラーが出力されている場合、このエラーに相当します。

<db2diag.log の出力例> ログを使い切ってしまっている場合

```
2002-06-25-08.59.43.116846   Instance:db2inst1   Node:000
PID:76916(db2agent (TESTDB))   Appid:*LOCAL.db2inst1.020624235616
data_protection  sqlpgrsp   Probe:20   Database:TESTDB
Warning: active log held by dirty pages.
Decrease softmax and/or increase num_iocleaners.
2002-06-25-08.59.43.386892   Instance:db2inst1   Node:000
PID:76916(db2agent (TESTDB))   Appid:*LOCAL.db2inst1.020624235616
data_protection  sqlpgrsp   Probe:50   Database:TESTDB
Log Full -- active log held by appl. handle 0
End this application by COMMIT, ROLLBACK or FORCE APPLICATION.
2002-06-25-08.59.43.661804   Instance:db2inst1   Node:000
PID:76916(db2agent (TESTDB))   Appid:*LOCAL.db2inst1.020624235616
data_protection  sqlpWriteLR   Probe:80   Database:TESTDB
DIA3609C Log file was full.
ZRC=0xFFFFD509
```

② ログの設定をデータベース構成パラメーターより確認

```
$ db2 get db cfg for <database-alias> | grep ログ
ログ・ファイルのサイズ (4KB)                    (LOGFILSIZ) = 1000
 1 次ログ・ファイル数                           (LOGPRIMARY) = 100
 2 次ログ・ファイル数                            (LOGSECOND) = 20
```

③ ②の結果を元に Logfilsz、Logprimary、Logsecond を適正な値に増加

④ ログ用ファイルシステムの使用可能領域確認

df コマンドによりログ用ファイルシステムの使用可能領域を確認してください。

```
$ df -k
<実行結果>
Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/lvname     1146880         0   100%_    2187    2%   /<log_path>
```

⑤ ③、④の結果より、以下の値に相当する十分なスペースがあるかどうかを確認してください。

((logprimary + logsecond) * (logfilsiz + 2) * 4096) + 8192 Byte

⑥ アプリケーション側で適切な COMMIT/ROLLBACK が発行されているかも確認してください。

3-1-1-5 SQL0968C(ファイルシステムがいっぱい)

【解説】

データベース領域(データベース・ディレクトリー、データベース・ログ・ファイル、または表スペース・コンテナーなど)に関連するファイル・システムのいずれかがあふれている可能性があります。

【補足】

レジストリー変数DB2_BLOCK_ON_LOG_DISK_FULLをONにすると、ログディスクフル時にエラーを返さず、ブロックします。5 分後にリトライを行うようになります。(DB2 V7.2 もしくはFixpak3 以降)。5-3-2-1 節ログディスクフル設定時の待ちを参照してください。

このエラーを発生する原因となる障害例のうち本書で対応方法を掲載しているものを以下に記載いたします。

[a]データベース構成パラメーターにおいて、LOGRETAIN=ON、USEREXIT=OFF の場合に、ログファイル用スペースを使い切ってしまった場合(想定例 1)

db2diag.log に以下のメッセージが表示された場合、このエラーに相当します。

下記と同様のエラーが発生している場合以下の<想定例 1>を参照してください。

<db2diag.log の出力例>

```
2002-07-12-13.36.56.547740   Instance:db2inst1   Node:000
PID:112506(db2loggr)   Appid:none
data_protection  sqlpgifl   Probe:30
DIA3612C Disk was full.
ZRC=0xFFFFD60C
```

[b]USEREXIT 設定時のアーカイブ先の容量不足により、ログファイルをコピーできず、アーカイブログが増加し続けログファイル用スペースを使い切ってしまった場合(想定例 2)

db2diag.log に以下のメッセージが表示された場合、このエラーに相当します。

下記と同様のエラーが発生している場合以下の<想定例 2>を参照してください。

<db2diag.log の出力例>

```
2002-06-25-08.56.17.263199   Instance:db2inst1   Node:000
PID:103462(db2cart)   Appid:none
data_protection  sqlpgart   Probe:490
User Exit returned error when archiving log file S0000142.LOG from /database/db2inst1/NOD
E0000/SQL00001/SQLOGDIR/ for database TESTDB, error code 28
2002-06-25-08.56.17.544864   Instance:db2inst1   Node:000
PID:103462(db2cart)   Appid:none
data_protection  sqlpgart   Probe:490
Error received from userexit program.
DB2 will not call userexit program for this database for 5 minutes.
2002-06-25-09.02.23.154975   Instance:db2inst1   Node:000
PID:158382(db2loggr)   Appid:none
data_protection  sqlpgmar   Probe:55
At least 5 minutes has passed since the last time an error was received when
archiving a log file.  Give it another try on log file S0000146.LOG
```

上記に当てはまらない場合は、以下の【対応方法(共通)】を順に実行し、原因の特定を行ってください。

【対応方法(共通)】

① db2diag.log より確認

上記[a]、[b]のどちらにも該当しない場合、その他のメッセージより判断可能な場合、対応する手順を実行してください。また以下の②により不足している領域を特定してください。

② 領域不足になったファイルシステムがあるか、特定してください。

次のチェックポイントを順に確認し、領域不足のファイルシステムが特定できたら、③に進んでください。

＜チェックポイント＞

1. df コマンドを実行し、使用率が100%または100%近くになっているファイルシステムがあるか、確認してください。もし、そのようなファイルシステムがあり、そのコマンドの実行に関連するファイルシステムであれば、そのファイルシステムで上記エラーが発生した可能性があります。

```
$ df -k | more
```

```
Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4            40960     21688   48%     2525    13% /
/dev/hd2          3293184     51012   99%    66165     9% /usr
/dev/hd9var         81920        68  100%      767     4% /var
/dev/hd3            32768     10692   68%      466     6% /tmp
/dev/hd1            49152     45032    9%      182     2% /home
/dev/lv_DB2DATA   1228800    605932   51%     6304     3% /db2/data
/dev/lv_DB2DATA2   499712         0  100%     3012     3% /db2/data2
```

2. OSのエラーログを参照してディスクやファイルシステム関連のエラーが出ているか確認し、可能であればエラーの発生箇所を特定してください。(1 章 2-3-1 節エラーログ参照)

③ ファイルシステム中の重要な初期情報の退避

以下のファイルは削除してしまわないように十分注意してください。

- システムコアファイル(1 章 2-3-3 節corefile参照)
- DB2 コアファイル(1 章 2-1-3 節DB2ダンプ・ファイル/DB2 コアファイル参照)
- DB2 トラップ・ファイル(1 章 2-1-4 節DB2 トラップ・ファイル参照)

退避するために以下の手順を実行してください。

$ cd <*path*>

$ mv <退避ファイル名> <退避先 PATH>

④ 対象の領域の空き容量を確保してください。

※どの程度空き容量が必要かは、状況によって異なります。

1. 不要なファイルを対象ファイルシステム上から移動、削除もしくは圧縮する。
2. 領域を拡張する。(OS 側のマニュアル参照)

以下にこのエラーを発生させる障害例とその対応方法を記載いたします。

<想定例1> 保存ログ方式かつ USEREXIT によりログの退避を行っていない場合に、ログファイル用スペース不足となってしまった場合、

【対応方法(個別)】

① ログの設定をデータベース構成パラメーターより確認

```
$ db2 get db cfg for <database-alias> | grep ログ
ログ・ファイルのサイズ (4KB)                    (LOGFILSIZ) = 1000
 1 次ログ・ファイル数                           (LOGPRIMARY) = 100
 2 次ログ・ファイル数                            (LOGSECOND) = 20
 ログ・ファイル用に変更されたパス               (NEWLOGPATH) =
 ログ・ファイルのパス                                 = /<log_path>
```

② ログ用ファイルシステムの使用可能領域確認

df コマンドによりログ用ファイルシステムの使用可能領域を確認してください。

```
$ df -k
```

<実行結果>

```
Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/lvname     1146880         0   100%_    2187    2%   /<log_path>
```

※ Free:0、%Used:100%になっていないか確認して下さい。

③ ①、②の結果より、以下の値に相当する十分なスペースがあるかどうかを計算してください。

((logprimary + logsecond) * (logfilsiz + 2) * 4096) + 8192 Byte

④ スペースが足りない場合空き領域を確保してください。

※どの程度空き容量が必要かは、状況によって異なります。

1. 不要なファイルを対象ファイルシステム上から移動、削除もしくは圧縮する。
2. 領域を拡張する。(OS 側のマニュアル参照)

⑤ USEREXIT によるログの退避も検討してください。

<想定例2> USEREXIT 設定時のアーカイブ先の容量不足によるアーカイブログの増加

アーカイブ先の容量不足によりアーカイブログの退避が出来なくなりアーカイブログが増加し続けログファイル用のスペースが不足してしまった場合

【対応方法(個別)】

① USEREXIT プログラムを使用してログの退避を自動化する場合は、データベース構成パラメータ USEREXIT を"ON"にする必要があります。正しく設定されているか確認してください。

＜USEREXIT 設定の確認例＞

```
$ db2 get db cfg for <DB 名> | grep USEREXIT
ロギング用ユーザー出口使用可能                  (USEREXIT) = ON
```

② USEREXIT プログラムが、適切なファイル名(パス名とファイル名)で存在し、かつ、適切なアクセスモードを持っているか、確認してください。

ファイル名： $INSTHOME/sqllib/adm/db2uext2

アクセスモード： インスタンスオーナーに読取り＋実行権限がある事

③ USEREXIT のログを参照し、何かエラーが出ていないかを確認してください。

1 USEREXIT.ERR からの判別

リターンコード(User Exit RC)を基に対応してください(下記リターンコード表参照)

USEREXIT.ERR は USEREXIT 内で設定しているパスに出力されます。

<USEREXIT.ERR 出力例> アーカイブ先の容量不足の場合

```
Time of Error:      Fri Jul 12 14:44:48 2002

Parameter Count:       8
Parameters Passed:
Database name:      TESTDB
Logfile name:       S0000013.LOG
Logfile path:       /home/db2inst1/test1_db2inst1/TESTDB/log/
Node number:        NODE0000
Operating system:   AIX
Release:            SQL07024
Request:            ARCHIVE
Audit Log File:     /home/db2inst1/test2_db2inst1/audit_error/ARCHIVE.LOG
System Call Parms: cp /home/db2inst1/test1_db2inst1/DB21DB1/log/S0000013.LOG
/home/db2inst1/test2_db2inst1/DB21DB1/NODE0000/S0000013.LOG
Media Type:         disk
User Exit RC:       28
```

2 ARCHIVE.LOG からの判別

リターンコード(User Exit RC)を基に対応してください(下記リターンコード表参照)

<ARCHIVE.LOG 出力例> アーカイブ先の容量不足の場合

```
Time Started:       Fri Jul 12 14:44:48 2002

Parameter Count:       8
Parameters Passed:
Database name:      DB21DB1
Logfile name:       S0000013.LOG
Logfile path:       /home/db2inst1/test1_db2inst1/DB21DB1/log/
Node number:        NODE0000
Operating system:   AIX
Release:            SQL07024
Request:            ARCHIVE
System Action:      ARCHIVE from /home/db2inst1/test1_db2inst1/DB21DB1/log/ file
S0000013.LOG to /home/db2inst1/test2_db2inst1/DB21DB1
Media Type:         disk
User Exit RC:       28              |||> ERROR <|||
Time Completed:     Fri Jul 12 14:44:48 2002
```

※リターンコード一覧表:

リターンコード	説明	注
0 RC_OK	正常終了	5 分以内にリトライ(繰り返しリトライ)
4 RC_RES	リソースアロケーションエラー	5 分以内にリトライ(繰り返しリトライ)
8 RC_OPATTN	オペレーターの介入が必要なエラー	5 分間保留後、最初の機会にリトライ
12 RC_HARDWARE	ハードウェアエラー	5 分間保留後、最初の機会にリトライ
16 RC_DEFECT	ソフトウェアエラー	5 分間保留後、最初の機会にリトライ
20 RC_OARM	パラメーターエラー	5 分間保留後、最初の機会にリトライ

24 RC_NOTFOUND	ファイルが発見できなかった	5 分間保留後、最初の機会にリトライ
28 RC_UNKNOWN	UNKNOWN エラー(上記以外のエラー)	5 分間保留後、最初の機会にリトライ
32 RC_OPCAN	ユーザによる終了	5 分間保留後、最初の機会にリトライ

④ アーカイブ先の領域の空き容量を確保してください。

※どの程度空き容量が必要かは、状況によって異なります。

1. 不要なファイルを対象ファイルシステム上から移動、削除もしくは圧縮する。
2. 領域を拡張する。(OS 側のマニュアル参照)

⑤ アーカイブログの保存方法・運用方法を再検討してください。

3-1-1-6 SQL1032N(データベースマネージャが起動されていない)

【解説】

start database manager コマンドが処理されていません。 SQL ステートメント、他のコマンドを発行する前に、このコマンドを処理する必要があります。

【対応方法】

以下のコマンドを実行することによりデータベースマネージャーが起動します。

```
$ db2start
```

3-1-1-7 SQL1042C(OSから予期しないシステムエラーが返された)

以下にまず行っていただきたい対応手順を記載いたします。

【対応方法(共通)】

クライアントあるいはサーバーの db2diag.log を見てエラーが生じているかどうかを調べます。ある程度見当が付くエラー内容であれば、その対応を実施してください。

以下に本節で紹介しておりますこのエラーを出力する原因となる障害例をリストいたします。

○ 修正パックのインストール後、インスタンスの更新を行っていない場合
○ 破損回復時にログファイルにエラーが発生した場合
○ インストール済み DB2 モジュールが破損、または、修正パック適用が失敗している場合
○ SMS 表スペースを含む Split ミラーDB をバックアップした場合

以下にそれぞれの対応を記載いたします。

<想定例1> 修正パックのインストール後、インスタンスの更新を行っていない場合

【解説】 修正パックやオペレーティング・システムのレベル変更が、DB2 インスタンス(データベースマネージャー)に識別される必要があります。

【対応方法(個別)】 DB2 の修正パックのインストール後かオペレーティングシステムのレベルの変更時に db2iupdt を実行してください。(root 権限が必要)

```
# /usr/opt/db2_08_01/instance/db2iupdt  -u  〈fenced_id〉  〈InstName〉
```

<想定例2> 破損回復時にログファイルにエラーが発生した場合

【解説】 db2diag.logに次のような項目があった場合です。(1 章 2-1-1 節診断ログ(db2diag.log)参照)

```
2002-06-24-18.15.07.687224   Instance:db2inst1   Node:000
PID: 126858(db2agent (DB21DB1))   Appid:*LOCAL.db2inst1.020624091044
buffer_pool_services   sqlbstartpools   Probe:0   Database:DB21DB1
Starting the database.
2002-06-24-18.15.08.340695   Instance:db2inst1   Node:000
PID: 126858(db2agent (DB21DB1))   Appid:*LOCAL.db2inst1.020624091044
data_protection        sqlpresr      Probe:0   Database:DB21DB1
Crash Recovery has been initiated.   【1】
2002-06-24-18.15.08.740917   Instance:db2inst1   Node:000
PID: 126858(db2agent (DB21DB1))   Appid:*LOCAL.db2inst1.020624091044
data_protection        sqlpgole      Probe:30   Database:TESTDB
A problem occurred while verifying a database log file S0000000.LOG
RC=ffffe60a   【2】
2002-06-24-18.15.09.351432   Instance:db2inst1   Node:000
PID: 126858(db2agent (TESTDB))   Appid:*LOCAL.db2inst1.020624091044
data_protection        sqlpgilt      Probe:101   Database:TESTDB
DiagData
0ae6 ffff
```

① 【1】はデータベースの破損回復が開始したことを表示しています。最後に使用されたときにデータベースが正常に停止しなかったことを表しております。

② 【2】ログファイルでエラーが発生したことを表しております。(1 章 2-1-1 節診断ログ(db2diag.log) <16 進コードの解釈>参照)。例の場合は、エラーコードは"e60a"になります。16 進数のエラーコードを得たら、それを 10 進形式に変換します。そのコードがメッセージ解説書 にある場合、それは SQL コードです。 エラー・コードの 10 進変換が SQL コードではない場合、戻りコードです。戻りコードのリストは、『問題判別の手引き 付録 A DB2 内部戻りコード』を 16 進数形式で参照してください。

③ 『問題判別の手引き』で戻りコードを調べると、ファイルが存在していないことがわかります。データベースの再始動および回復には、データベースが停止した時に使用中だったすべてのログ・ファイルが必要となります。この db2diag.log 項目はログ・ファイル S0000000.LOG が予期していた場所に見つからないことを示しています。

【対応方法(個別)】

この問題を解決する最もよい方法は、バックアップから復元することです。(1 章 2-1-4-2 節バージョン回復参照) この例では、ログファイル名がS0000000.LOGであるため、データベースは循環ログ方式か、あるいは、保存ログ方式だが比較的データベースを作成した直後であると推定されます。

<想定例3> インストール済み DB2 モジュールが破損、または、修正パック適用が失敗している場合

【対応方法(個別)】

同一サーバ上の他の DB2 インスタンスについても同様のエラーが発生するかを確認してください。

○全インスタンスで同様にエラーが発生する場合

⇒ DB2 インストール先の障害によって DB2 モジュールが破損している可能性が考えられます。この

ケースには、物理的な破損だけでなく、論理的な破損(ファイルシステムの不整合、誤った削除や上書きなど)も含まれます。

次のようなチェックを行い、DB2 モジュールが破損していると思われる場合は、再インストールを試みてください。

① エラーメッセージの内容確認

OSエラーログ(errpt コマンド, syslog etc.)や DB2DIAG.LOG を参照し、破損ファイル関連の情報が出ていないかを確認してください。ある程度見当の付くエラーメッセージが出ている場合は、その内容に応じた対応を行ってください。

② ディスクおよびファイルシステムの状態確認

ディスクチェックおよび、ファイルシステムチェックで、何らかの異常がないかどうか確認してください。(OS 側のマニュアル参照)異常がある場合は、回復に必要な処置を行ない、DB2 を再インストールしてください。

③ インストールもしくは修正パック適用状態の確認

```
$ lslpp -ch 'db2*' | more
```

を実行し、ステータスが“COMPLETE”以外のファイルセットがあるかを確認してください。特に、ステータスが“BROKEN”になっていないかがポイントです。

<想定例4> SMS 表スペースを含む Split ミラーDB を BACKUP コマンドでバックアップした場合

【解説】

V7.2 以降では、Split Mirror イメージが db2inidb コマンドによって ROLLFORWARD 状態になっているものについては、データベースのフルバックアップが取得できるようになりました。ただし、DMS 表スペースのみで作られたデータベースのフルバックアップのみサポートされます。SMS 表スペースを含むバックアップは失敗します。

＜失敗時の db2diag.log のメッセージ出力例＞

```
PID:45318(db2agent (TESTDB))   Appid:*LOCAL.udb32v7.010524132140
database_utilities  buildAppTblsp   Probe:37   Database:TESTDB
Backing up of a split mirrored database containing SMS Tablespaces is unsupported
```

3-1-1-8 SQL1224N(データベースエージェントが開始できない、あるいは強制終了した)

データベース・エージェントが、要求を処理するために開始できなかったか、あるいはデータベース・システムの遮断または強制コマンドにより終了しました。

以下にこのエラーを出力する原因となる障害例とその対応方法を記載いたします。

<想定例1> AIX の Shared Memory Segment 数の制限を越えてしまった場合

【解説】

AIX にはプロセス毎に最大 11 個までの Shared Memory Segment しかアロケートできないという制限があります。DB2 のデータベースに対するローカル Connection 1 個について1つの Shared MemorySegment がアロケートされます。このため、ローカル・アプリケーションの1プロセスから、同時に10以上のデータベ

ース接続を行いたい場合には、AIX の Shared Memory Segment 数の制限にかかり、AIX は ErrorNo 24(EMFILE)を返し、DB2 はアプリケーションに SQL1224N を返します。 例えばアプリケーション・サーバーがローカルの DB2 データベースを使用するケースで発生することがあります。

【対応方法】

ローカル・アプリケーションの1プロセスから、同時に10以上のデータベース接続を行いたい場合には、ローカル・データベースをリモート・ノードに存在するデータベースとしてカタログして接続して下さい。この時、ホスト名または IP アドレスの指定に loopback を使用すると、ネットワーク負荷の軽減になります。loopback については、/etc/hosts に、以下の指定がされている必要があります。

```
#vi /etc/hosts
 127.0.0.1    loopback localhost
```

<方法1>アプリケーションから接続するデータベース名を変更可能な場合

以下の手順で loopback を使用したリモート・ノードをカタログし、そこにデータベースが存在するように、別名でカタログしてください。

① サーバーローカルのノードディレクトリーに TCP/IP ノードを追加

```
$ db2 catalog tcpip node <node_s> remote <hostname> server <service_name>
```

<*node_s*>:ノード名

<*hostname*>:サーバーのホスト名または IP アドレス ＜＝＝ loopback を指定

<*service_name*>:サーバーインスタンスのサービス名またはポート番号

② サーバーローカルにリモート・ノードをデータベース別名でカタログ

```
$ db2 catalog database <database-name> as <database-alias> at node <node_s>
```

<*database-name*>:データベース名

<*database-alias*>:データベース別名。CONNECT 時にはこの名前を指定する。

<*node_s*>:カタログしたノード名

<方法2>アプリケーションから接続するデータベース名を変更不可能な場合

クライアント・インスタンス db2inst_c を作成しサーバー・インスタンス db2inst_s に対し、loopback を使用したリモート・ノードをデータベース本名でカタログしてください。

①サーバーローカルにクライアント・インスタンスを作成

```
# /usr/opt/db2_08_01/instance/db2icrt -u fencename db2inst_c
```

②クライアント・インスタンスのノードディレクトリーに TCP/IP ノードを追加

```
$ db2 catalog tcpip node <node_c> remote <hostname> server <service_name>
```

<*node_c*>:ノード名

<*hostname*>:サーバーのホスト名または IP アドレス ＜＝＝ loopback を指定

<*service_name*>:サーバーインスタンスのサービス名またはポート番号

③クライアント・インスタンスのリモート・ノードにデータベース本名でカタログ

```
$ db2 catalog database <database-alias> as <database-name> at node <node_s>
```

<*database-name*>:データベース名

<node_c>:カタログしたノード名

<方法 3> EXTSHM 環境変数を用いる方法(v7.1+FP3 or v7.2 以降)

①$ export EXTSHM=ON

②$ db2set DB2ENVLIST=EXTSHM

③$ db2start

【補足】

SQL1224N は接続の強制終了によっても発生します。強制終了には、ユーザーからの Force Application コマンドの他、DB2 が何かしらのエラーでインスタンスを強制終了させる場合もあります

3-1-1-9 SQL30081N (通信エラーが検出された)

【対応方法】

<サーバー側>

⑤ 通信サブシステムまたはネットワーク・エラーが起きていないか確認する。

$ errpt

$ netstat -i

[a]ネットワークインタフェースは全て使用可能であるかを確認してください。

インタフェースに*印がついているときはそれは使用不可能状態を表します。

[b]サーバー側のネットワークトラフィックが上がっていないかを確認してください。

Oerrs 欄が Opkts の 1%より大きい時はインターフェースの負荷が大きすぎます。

⑥ サーバーのデータベース・マネージャーが正常に始動されているかを確認

サーバーでのデータベース・マネージャーの始動処理にて SQL5043 が返された場合は、DB2DIAG.LOG をチェックしてください。

⑦ 以下の設定が適切に行われているか確認(TCP/IP 接続の場合)

レジストリー変数(DB2COMM=TCPIP)

確認 $ db2set DB2COMM

設定 $ db2set DB2COMM =TCPIP

⑧ /etc/hosts、DNS の設定確認

クライアントからサーバーに ping でパケットが到達するか確認する。

$ ping <*hostname*>

$ ping <*ip-address*>

IP アドレスでは ping が到達するのに、ホスト名では到達しない場合、/etc/hosts の設定や DNS サーバーが稼動しているかどうかを確認する

⑨ データベースマネージャー構成パラメーター、/etc/services の設定確認

・サービス名(SVCENAME)の指定

確認 $ db2 get dbm cfg | grep SVCENAME

TCP/IP Service name (SVCENAME) = <*service_name*>

もし、SVCENAME が設定されていない場合は任意の名前を設定する。

設定　　$ db2 update dbm cfg using SVCENAME <*service_name*>

SVCENAME が設定されている場合、その SVCENAME が/etc/services 内に登録されているか確認し、登録されていなければ追記します。

$ grep <*service_name*> /etc/services

※データベース・マネージャー構成パラメーターを更新した場合は、変更を反映するために、データベース・マネージャーの停止と再始動を行ってください。

<クライアント側>

通信サブシステムまたはネットワーク・エラーが起きていないか確認してください。

3-1-1-10 CLI/ODBC/JDBC 関連(SQL0104N /SQL0204N / SQL0206N/ SQL1003N/ CLI0637E/)

以下に本節で紹介する障害例をリストいたします。

○ SQL1003N (サーバーへの接続時、CLI/ODBC/JDBC アプリケーションが失敗) (想定例 1)

○ SQL0204N / SQL0206N(ODBC アプリケーションで作成された大文字小文字混合の名前が原因で障害が発生(想定例 2)

○ CLI0637E/SQL0104N (JDBC 経由の Export が失敗) (想定例 3)

<想定例1> SQL1003N (サーバーへの接続時、CLI/ODBC/JDBC アプリケーションが失敗)

【解説】

DB2 サーバーへの接続に、Microsoft Query,Microsoft Access,またはロータス アプローチなどのアプリケーションを使用時、SQL1003N メッセージを受け取るか、一般保護障害（GPF）が発生する。

【対応方法】

以下の 3 点を確認してください。

- DB2 CLI/ODBC ドライバーが正しく構成されていること確認

  DB2 クライアント構成アシスタントを使用することによって、ドライバーを構成する事ができます。

  ① 構成するデータベース別名を選択する。

  ② 「プロパティー（Properties）」をクリックして、「データベースのプロパティー（Database Properties）」ウィンドウを表示する。

  ③ 「設走（Settings）」をクリックして、「CLI/ODBC 設定（CLI/ODBC Settings）」ウィンドウを表示する。

  ④ 「拡張（Advanced）」をクリックする。構成キーワードを設定できるウィンドウが表示されます。特に、以下のことを確認して下さい。

  - ラージ・オブジェクト2進（LOB）データにアクセスしている場合、LOMGDATACOMPAT＝1
  - 表名に下線（__）がある場合、UNDERSCORE＝0

  あるいは、db2cli.ini ファイルを編集し、接続しているデータベースの別名の後にリストされているパラメーターのリストにキーワードがあるか確認してください。キーワードがそこにない場合、ファイルを編集し、別の行にそれらを追加して下さい。

  ※CLI ドライバー設定(db2cli.ini)をカスタマイズすると、一部のアプリケーションに障害が発生する可能性があります。これには、DB2 Java ベースのユーティリティーの一部も含まれます。それぞ

れの CLI/ODBC/JDBC ベースのアプリケーションにて使用する、異なるデータベース別名を作成して、各アプリケーションについてドライバー設定を独立して変更できるようにするとよいでしょう。

- DB2 CLI/ODBC ドライバーの最適化

  DB2 CLI/ODBC ドライバーは Lotus Approach、Microsoft Access、又は Visual Basic を使用される場合にはそれぞれのプログラム用にご提供している最適化設定をご利用になることをお勧めします。DB2 クライアント構成アシスタントを使用することによってドライバーを最適化することが出来ます。

  ① 構成する DB2 データベース別名を選択する

  ② 「プロパティー（Properties）」をクリックして、「データベースのプロパティー（Database Properties）」ウィンドウを表示する。

  ③ 「設走（Settings）」をクリックして、「CLI/ODBC 設定（CLI/ODBC Settings）」ウィンドウを表示する。

  ④ 「最適化（Optimize）」押しボタンを選択する。ウィンドウが表示され、この上部に表示されるアプリケーションのうち 1 つを選択します。このアプリケーションについて、事前定義されたバッチ値及び構成オプションを使用して CLI ドライバーが自動的に構成されています。

- Microsoft ODBC ドライバー・マネージャーファイルを更新

  一般保護例外(GPF)が起きている場合は Microsoft ODBC ドライバー・マネージャーファイルを更新する事で解決できる場合があります。(詳細は ODBC ドラーバー・マネージャーのマニュアルを参照ください。)

<想定例2> SQL0204N / SQL0206N（ ODBC アプリケーションで作成された大文字小文字混合の名前が原因で障害が発生）

【解説】

DB2 では、表名と列名を含め、引用符で囲まれていないオブジェクト名は小文字は大文字に変換されて解釈されます。名前に小文字を含めたい場合は、二重引用符に囲んで指定しなければなりません。そうしないと、DB2 はその名前を大文字に変換します。これは、CREATE TABLE 及び DROP TABLE ステートメントだけでなく、SELECT ステートメントを含む、すべての SQL ステートメントに当てはまります。一方、ロータスアプローチを含む、多くの ODBC アプリケーションは、表名及び列名の大文字小文字の区別を保持します。ODBC アプリケーションで表を作成する時、表名と列名を大文字小文字混合で指定すると、次のような SQL ステートメントが DB2 に送られます。

CREATE TABLE “test1” (“col1” CHAR(5))

この場合、表名と列名がそれぞれ二重引用符で囲まれているので、DB2 は名前を小文字で保持します。名前を二重引用符で囲まずにこの表にアクセスしようとしても成功しません。

コントロール・センターでは、大文字小文字が混在する名前オブジェクトは、二重引用符で囲まれ表示されます。

<症状 1> SQL0204N("<*name*>" は未定義の名前です。)

特定の種類のアプリケーション（たとえば、ロータス アプローチなどの ODBC アプリケーション）で小文字を含めたオブジェクトを作成し、そのあとで別の方式で（たとえば、DB2 コマンド行プロセッサーによっ

て)これにアクセスした場合、表へのアクセスに問題の起こることがあります。たとえば、SELECT * FROM tablel などの SQL ステートメントに対して、このようなエラー・メッセージが返されることがあります。

たとえ表が存在しても、このエラー・メッセージが返されます。コントロール・センターの表のリスト、又は LIST TABLES コマンドの出力には示されます。

このエラーは、逆にオブジェクトの作成時には大文字だけで名前を付けていたのに、オブジェクト名を指定するときには大文字小文字混在で、引用符を使った場合にも生じます。たとえば、ステートメント

SELECT * FROM “Org”

もエラーを返します。

<症状 2> SQL0206N(使用されているコンテキストで、"<*name*>" は無効)

SELECT columnl from table2 などの SQL ステートメントに対して、このようなエラー・メッセージが返されることがあります。

このエラー・メッセージは、以下の条件が満たされる場合であっても戻されます。

①列”column1”が存在し、コントロール・センターに表示される。又は SQL ステートメント

SELECT tabname, colname FROM SYSCAT.COLUMNS WHERE tabname= ‘table2’

が、”column1”列に関する適切な情報を返す。

②SQL ステートメント

SELECT * FROM table2

が”column1”列を含め、全ての列を返す。

【対応方法】

ODBC アプリケーションを使用して DB2 表を作成する場合は、表名と列名を大文字で指定して下さい。そうすると、二重引用符の有無に関係なくこれらの表又は列にアクセス出来るようなります。表及び列を大文字小文字混合で処理することが必要な場合は、オブジェクト名を全て二重引用符で囲んで下さい。たとえば、以下のように表を作成すると、そのあとの SELECT ステートメントは両方とも成功します。

<入力例>

```
$ db2
db2 => create table "User1"."Table1" ("column1" CHAR(5))
db2 => SELECT * FROM "User"."Table1"
db2 => SELECT "column1" FROM "User1"."Table1"
```

<想定例3> CLI0637E/SQL0104N (JDBC 経由の Export が失敗)

【解説】

JDBC 経由で Statement の execute(sql)メソッドに EXPORT コマンドを渡すと、以下のようなメッセージが出力されます。

① SQL0104N "BEGIN-OF-STATEMENT" に続いて予期しない字句 "EXPORT TO X:/XXXX/" が見つかりました。入力が予想される字句には "<create_view>" が含まれている可能性があります。

② CLI0637E “QUERY が見つかりません。”

【対応方法】

DB2 エクスポートなどのコマンドをプログラムで使用することはできません。API を使用してください。もし、プログラムでエクスポートを行う場合は EXPORT API をコールしてエクスポートを行います。詳細は『データ移動ユーティリティー手引きおよび解説書 第1章 エクスポート エクスポートセッション例』内の「APIの例」を参照してください。sqluexpr のインターフェースに関しては「エクスポートAPI」を参照してください。コーディング例は$INSTHOME/sqllib/samples/c に impexp.sqc が導入されていますので参考にしてください。

3-1-1-11 ロード保留状態時のエラー(SQL2048N/SQL3805N)

【解説】

表スペースをロード保留状態にしたアプリケーションが何らかの形で存在する場合には、他のアプリケーションから保留状態を解除することはできません。その場合に、以下の操作を実行するとこのエラーが発生します。

- QUIESCE TABLESPACES FOR TABLE コマンドを行った場合

  $ db2 quiesce tablespaces for table 〈*table-name*〉 reset

  SQL2048N オブジェクト"〈*table-name*〉 "のアクセス中に、エラーが発生しました。理由コード: "8"。

- ロード保留の解除を試みた場合(1 章 1-4-3 節LOAD)

  $ db2 “load from 〈*file-name*〉 of 〈*file-type*〉 terminate into 〈*table-neme*〉”

  SQL3805N アプリケーション、または指定された表の 1 つ以上の表スペースの状態が、loadapi アクションまたは quiescemode "2" を禁止しています。理由コード = "1"。

この場合、すべてのアプリケーションが切断された状態で、あらためて他のアプリケーションからコマンドを発行すれば、保留状態を解除できます。以下の手順にて対応を行ってください。

【対応方法】

① 以下のコマンドを実行し、アプリケーションの接続状況を確認します。

$ ps -ef | grep 〈*instance-name*〉 | grep 〈*database-alias*〉

を実行した場合、自分以外に一プロセス存在し、

$ ps -ef | grep 〈*instance-name*〉 | grep db2bp

を実行した場合、自分以外に存在しないことを確認します。

※プロセスが全然残っていないと、表スペース保留状態は上記の【解説】に記載したコマンドで解除できるはずです。

② 表スペースの状態を確認します。

$ db2 list tablespaces

```
状況                                 = 0x000c
 詳しい説明:
   静止: EXCLUSIVE
   ロード保留中
```

この状態が表示された場合、以下の手順で、ロード保留状態が解除できるところまでを実行ください。

③ QUIESCE TABLESPACES FOR TABLEコマンドを実行します。

```
$ db2 quiesce tablespaces for table <table-name> reset
```

④ ロード保留の解除を試みます。(1章1-4-3節LOAD)

⑤ TERMINATEコマンドを実行します。

```
$ db2 terminate
```

⑥ ③、④を再度実行します。

⑦ 全アプリケーションの切断を行います。

```
$ db2 force application all
```

⑧ ③、④を再度実行します。

⑨ データベースマネージャーの停止/開始を実行します。

```
$ db2stop
$ db2start
```

⑩ ④を再度実行します。

3-1-1-12 SQL6048N/SQL1032N (START または STOP DATABASE MANAGER 処理中に通信エラーが発生しました)

【解説】

HACMP などのクラスター・ソフトウェアを用い、アクティブ・スタンバイの構成を構築してある場合、スタンバイ機へのテークオーバー後、スタンバイ機のホスト名と db2nodes.cfg ファイル内のホスト名が異なると、db2start がこのエラーにより失敗します。

【対応方法】

以下のいずれかの方法を検討します。

① 正しいホスト名の db2nodes.cfg を予め別途用意しておき、それで置き換えてから db2start を実行します。

② テークオーバー後の db2 起動時に以下のコマンドを実行します。

```
$ db2start DBPARTITIONNUM [ノード番号] restart HOSTNAME [ホスト名]
```

[ホスト名]にはスタンバイ・マシンのhostnameコマンドで戻る値を割り当てます。これにより、$INSTHOME/sqllib/db2nodes.cfgが[ホスト名]に合わせて変更されます。

③ クラスターSWでDB2のサービス IP アドレスをテークオーバーする場合、サービス IP アドレスに対応した仮想ホスト名を db2nodes.cfg に記述すると、テークオーバー後に db2nodes.cfg は書き換える必要なく、db2start コマンドが成功します。

【参考】

DB2 V8 の ESE では、従来(V7.2 まで)の EEE(Enterprise Extended Edition)のコンポーネントが EE(Enterprise Edition)に統合され、SQLLIB ディレクトリーの下のインスタンス・ディレクトリーの下に、db2nodes.cfg ファイルが作成され、使用されるようになりました。

4 エラーコードが表示されない場合の対応方法(H/W、OS レベルに問題がある可能性が高い場合)

2-3 節エラーコードは戻らないが、処理のレスポンスがない場合での問題判別で、H/WまたはOSレベルに問題があると判断された場合、以下の対応を行ってください。詳細はH/WあるいはOSの問題判別手順書を参照ください。

4-1 HW に問題がある可能性が高い場合

【対応方法】

①詳細レポート(errpt –a)を確認してください。

```
$ errpt -a
```

```
---------------------------------------------------------------------
ラベル: STOK_WIRE_FAULT
識別子: 2A9F5252

日付／時間:       Mon Jun 10 08:12:55
順序番号:         1927
マシン ID:        000D19904C00
ノード ID:        vagi08
クラス:           H
タイプ:           PERM
資源名:           tok0
資源クラス:       adapter
資源タイプ:       14101800
ロケーション:     20-58

説明
ワイヤ失敗

考えられる原因
TOKEN-RING ローブ

障害の原因
ローカル・ローブ・ケーブル
ケーブルが外れているか、または欠陥があります
```

②db2diag.log を確認してください。

```
$ view $INSTHOME/sqllib/db2dump/db2diag.log
```

③対応不可能である場合、サポートセンターに連絡してください。

連絡時に必要な情報

- 本章第 1 節に記述された情報(特に 1-[d])
- 詳細レポート(errpt –a)の情報

4-2 OS に問題がある可能性が高い場合

【対応方法】

①取得できる場合、詳細レポートを確認してください。

```
$ errpt -a
```

②OS 側の問題判別手順書が存在する場合参照してください。

③db2diag.log を確認してください。

```
$ view  $INSTHOME/sqllib/db2dump/db2diag.log
```

④どのコマンドも実行不可能かつ、ディスクのアクセスが行われていないように見える場合、十分注意した上で、マシンのリブートを行ってください。

⑤対応不可能である場合、サポートセンターに連絡してください。

連絡時に必要な情報

- 本章第 1 節に記述された情報(特に 1-[e])
- 詳細レポート(errpt –a)の情報

4-3 空き領域不足の場合

①領域不足になったファイルシステムを特定してください。

②対象のファイルシステムの空き容量を確保してください。

※どの程度空き容量が必要かは、状況によって異なります。

1. 不要なファイルを対象デバイス上から削除もしくは圧縮してください。(ファイルシステムの場合)
2. ファイルシステムの領域を拡張してください
3. 必要であれば物理 Disk を追加してください

5 エラーコードが表示されない場合の対応方法(DB2 に問題がある可能性が高い場合)

5-1 インスタンスダウンの可能性が高い場合の対応方法

2-3 節エラーコードは戻らないが、処理のレスポンスがない場合での問題判別で、DB2インスタンスのプロセス(またはスレッド)本体が停止している状態と判断された場合、まず以下の【対応方法(共通)】の①を行ってください。

【対応方法】

① 原因を究明します。db2diag.log を確認してください。
不明な場合、下記の<想定例 1、2、3>のいずれに当てはまるかをまず確認してください。
→<想定例 1>を参照してください。

② インスタンスのリスタートを行ってください。
障害等のアプリケーション中断によるデータベースの矛盾を除き、使用不可能な状態を解除する為に、自動又は手動の破損回復方法を選択し実行して下さい。
<リスタート手順>
自動の場合: (条件)データベース構成パラメータ AUTORESTART が YES の場合

```
$ db2start
  $ db2 connect to <database-alias>
```

手動の場合: (条件)データベース構成パラメータ AUTORESTART が NO の場合

```
$ db2start
$ db2 restart database <database-alias>
```

※詳細は1章 2-1-1 破損回復を参照して下さい。

③ インスタンスリスタート実行時にエラーが発生していないか、破損回復が完了したかを db2diag.log により確認を行ってください。

④ DB へ接続できたかを確認してください。
※破損回復が完了しない限りデータベースへの接続が出来ませんので注意して下さい。

```
$ db2 connect to
```

⑤ 表スペースに障害が発生していないかの確認をしてください。

```
$ db2 list tablespaces
```

【補足】

誤ったシグナルの送信がインスタンスダウンを発生させる場合もございます。kill -9 コマンドの使用は避けてください。

上記の【対応方法】の①で db2diag.log の内容からは不明な場合、以下の障害例を参考に対応を実施してください。

<想定例1> ulimit の STACK+DATA が 256MB 超えている場合

【対応方法(個別)】

①ulimit の設定の見直し

AIX では、1セグメント内に Agent Privare Memory がとられ、32ビット・メモリー・モデルでは、1セグメントは256MB です。従って、Agent Private Memory に取られる要素の合計が256MBを越えることはできませんが、ulimit でこの制限を越すよう許容していたため、制限を越して使ったことがインスタンスダウンを誘発するケースがあります。そのため、Agent Private Memory の使用可能領域を広げるために、全ての DB2 ユーザーID について、 AIX の ulimit(/etc/security/limits)の data と stack の設定を広げる際には、 data + stack =< 256MB の範囲で、data の領域を広げてください。

ただし、使用可能領域を広げても、Agent Private Memory の使用量がこの値を越えた場合には、何らかのメモリー不足エラーが発生することは避けられません。

※Agent Private Memory にとられる内容を設定する構成パラメーターは以下です。

applheapsz、sortheap（Private sort の場合）、stmtheap、query_heap_sz、java_heap_sz、udf_mem_sz
stat_heap_sz、drda_heap_sz、agent_stack_sz

（設定例:この場合、Agent Private Memory の合計の上限は240MB）

1. 必ず/etc/security/limits ファイルのバックアップを取得
2. /etc/security/limits の内容の変更

```
$ vi /etc/security/limits
  -------以下を追記---------
<user-name> :
data=447519 block (512 byte block)   （デフォルト値は 262144 block）
stack=32767 block (512 byte block)   （デフォルト値は 65536 block）
--------------------------------
```

※制限なくメモリーを使用することを避けるために、unlimited には設定しない。

②上記の【対応方法(共通)】の②以降を参照

【補足】 V8.1 から、インスタンス作成時に、ulimit を以下のように自動的に設定変更します。

```
data=491519
stack=32767
```

<想定例2> データページの破損

【解説】

データ・ページの破損の形態には、ページ内のヘッダーの制御情報やポインターが壊れている、整合性チェックのデータが壊れている、ページ内のデータが別のデータで上書きされて壊れている、などのケースがあります。ページ破損が起こると DB2 はデータベースを Bad とマークし、DB2 インスタンスをシャットダウンしようとする事もあります。

【対応方法】

① db2diag.log にメッセージがでていないか確認してください。db2diag.log に以下のようなメッセージがでている場合はデータページ破損の可能性が高いです。

```
2002-06-24-12.48.12.380882    Instance:tx7ee    Node:000
PID:159196(db2agent (SAMPLE))    Appid:*LOCAL.tx7ee.020624034555
```

```
buffer_pool_services  sqlbrdpg   Probe:1146   Database:SAMPLE
DIA3847C An error occurred in a database page header.

ZRC=0xFFFFF136

2002-06-24-12.48.12.456208   Instance:tx7ee   Node:000
PID:159196(db2agent (SAMPLE))   Appid:*LOCAL.tx7ee.020624034555
buffer_pool_services  sqlbrdpg   Probe:1146   Database:SAMPLE

 Obj={pool:2;obj:14;type:0} State=x0 Page=0 Cont=0 Offset=0 BlkSize=12
BadPage

Data Title:SQLB_OBJECT_DESC PID:159196 Node:000
0002 000e 0002 000e 0000 0000 0000 0000       ................
0000 0000 0000 0000 0000 0000 0000 0000       ................
0000 0000 0000 0101 0000 0000 0000 0000       ................
0000 1000 0000 0020 0000 0001 000e 0002       ....... ........
400a 3118 4010 c030                           @.1.@.タ0

以下省略
```

上記の db2diag.log では、pool:2 より表スペース ID は 2、obj:14 よりオブジェクト ID は 14、type:0 よりオブジェクトのタイプは表(type:1 なら索引)であることがわかります。

② (オプション)db2dart コマンドで、①で検出された表のデータページ保全性を確認してください。db2dart を行う際には、インスタンスが停止していることを確認の上行ってください。

```
$ db2dart SAMPLE /T /TSI 2 /OI 14
  Table inspection start: TX7EE.STAFF2

      Data inspection phase start. Data obj: 14  In pool: 2
      Error: BPS check read error for pool page 0, from object ID 14, pool 2,
      Error: BPS Header object ID incorrect. Expecting 14, Found 15.
      Error: BPS Header problems found
      Error: in page 0, pool page 0, of Object 14, in tablespace 2.
      Error: Page data will be dumped to report.

        000       *0030 0FD0 0000 00C4 0300 0000 00BC 2552*   *.0.............R*

        010       *0000 0000 000F 0002 627D 7267 0000 0000*   *........b.rg....*
```

上記内容により BPS Header が破損していることがわかります。※出力例は、障害を意図的に起こすためにヘッダーの内容を書き換えているため、このようなエラーになっています。)

この場合は、データベースのバックアップ・イメージを戻し、ロールフォワード回復を行って回復させることにより復旧させます。(1 章 1-4-2 節バージョン回復参照)データベースのバックアップ・イメージが存在しない場合は、必要な資料を取得し(2 章 8-1 節サポートセンター連絡時に取得すべき情報参照)、サポートセンターに連絡してください。

<想定例3> 索引ページの破損

索引ページの破損の形態には、索引ページ内のヘッダーの制御情報が壊れている、整合性チェックの

データが壊れている、索引内のデータが別のデータで上書きされて壊れている、などのケースがあります。索引ページ破損が起こるとデータページの場合と同じように、DB2 はデータベースを Bad とマークし、DB2 インスタンスをシャットダウンしようとする事があります。

【対応方法】

① まず db2diag.log にメッセージがでていないか確認してください。db2diag.log に以下のようなメッセージがでている場合は索引ページ破損の可能性が高いです。

```
String Title:Index(es) need to be rebuilt. PID:89892 Node:000
Table(2:10)=DB2INST1    .EMP_RESUME

2002-06-26-16.55.46.376168   Instance:db2inst1   Node:000
PID:48236(db2agent (SAMPLE))   Appid:*LOCAL.db2inst1.020626075538
buffer_pool_services  sqlbrdpg   Probe:1143   Database:SAMPLE
DIA3700C A bad page was encountered.

ZRC=0xFFFFE101

2002-06-26-16.55.46.670837   Instance:db2inst1   Node:000
PID:48236(db2agent (SAMPLE))   Appid:*LOCAL.db2inst1.020626075538
buffer_pool_services  sqlbrdpg   Probe:1143   Database:SAMPLE

 Obj={pool:2;obj:10;type:1} State=x5 Page=0 Cont=0 Offset=0 BlkSize=12
BadPage

Data Title:SQLB_OBJECT_DESC PID:48236 Node:000
0002 000a 0002 000a 0100 0000 0000 0000       ................
0000 0000 0000 0000 0000 0000 0000 0000       ................
0000 0000 0000 0001 0000 0005 0000 0000       ................
0000 1000 0000 0020 0000 0001 010a 0002       ....... ........
400a 3118 4010 c030                           @.1.@.タ0
```

上記の db2diag.log では、pool:2 より表スペース ID は 2、obj:10 よりオブジェクト ID は 10、type:1 よりオブジェクトのタイプは索引(type:0 なら表)であることがわかります。

② (オプション)db2dart コマンドで、データページ破損を確認。db2dart を行う際には、データベースが停止していることを確認の上行ってください。

```
db2dart sample /T /TSI 2 /OI 10
      ______                  DART                  ______

  D a t a b a s e   A n a l y s i s   a n d   R e p o r t i n g   T o o l

                          IBM   DB2   6000
_____________________________________________________________________________

_____________________________________________________________________________
-----------------------------------------------------------------------------
  Action option: DI
Table-object-ID: 10; Tablespace-ID: 2; First-page: 0; Number-pages: 3; Verbose: y

Connecting to Buffer Pool Services...

   Index object report phase start.
```

```
Error: BPS check read error for pool page 0, from object ID 10, pool 2,
Error: BPS Header Revision Number (3) invalid, DB2DART processing will assume version 5,
Error: BPS Header BEGOFF (12288) incorrect, DB2DART processing will assume value of 48,
Error: BPS Header DATLEN (53263) incorrect, DB2DART processing will assume value of 4048,
       in pool page 0, object page 0, from object ID 10, pool 2.
```

上記内容により BPS Header が破損していることがわかります。

この場合は、データベースのバックアップ・イメージを戻し、ロールフォワード回復を行って回復させることにより復旧させます。(1 章 1-4-2 節バージョン回復参照)データベースのバックアップ・イメージが存在しない場合は、必要な資料を取得し(2 章 8-1 節サポートセンター連絡時に取得すべき情報参照)、サポートセンターに連絡してください。

5-2 処理中の可能性が高い場合の対応方法

2-3 節エラーコードは戻らないが、処理のレスポンスがない場合での問題判別で、処理は進行中である(CPUは使用している)が、通常時よりも非常に進み方が遅い、あるいは、同じ処理を延々と繰り返す状態で、そのまま放置していても、その状態から脱しないと判断された場合、以下の各項のどの現象に該当するかを特定し、対応を行ってください。

5-2-1 特定の SQL でレスポンスがない場合

以下に本節で紹介しておりますこのエラーを出力する原因となる障害例をリストいたします。

○ アクセスパスが不適切な場合

○ 表の断片化によるアクセス効率の低下

○ アプリケーションの自動再バインド発生の場合

以下にそれぞれの対応を記載いたします。

<想定例1> RUNSTATS の実行し忘れによる不適切なアクセスパス

【解説】

表のデータ内容が大幅に変わっているのに RUNSTATS を実行していないと DB2 オプティマイザーが適切なアクセスパスを選択できないことがあります。 また、新規に索引を作成後に RUNSTATS を行っていないと本来使われるべき索引が使用されないこともあり得ます。アクセスパスが適切か、意図とおりの変更がされたかの確認は、Explain 情報を取得し、以前のデータとの比較を行うことにより可能です。なお、静的 SQL ならばアクセスパスの更新には RUNSTATS 後に BIND/REBIND を行ってください。

【対応方法】

①Explain 情報の比較を行うために現在のアクセスパス情報を取得しください。

<静的 SQL の場合>

以下のコマンドを実行してください。

```
$ db2expln -d <database-alias> -p % -c % -s 0 -o <outputfile>
```

<動的 SQL の場合>

以下のコマンドを実行してください。

```
$ dynexpln -d <database-alias> -f <intputfile> -o <outputfile>
```

②RUNSTATSを実行してください(1 章 4-6 節RUNSTATS参照)

RUNSTATS は表内のデータの情報を収集し統計情報として保管します。その統計情報に基づき BIND 処理時にアクセスパスが決定されます。

③静的SQLの場合、REBINDを実行してください(1 章 4-8 節REBIND参照)

アプリケーションからデータベースへアクセスする場合の手法として、静的 SQL と動的 SQL に大別されます。静的 SQL では、実行時以前にどのようにデータをアクセスするか(アクセスパスまたはアクセス経路といいます)を決めておき、それをシステム・カタログに保存します。これを行うのが BIND/REBIND です。動的 SQL では、実行時にアクセスパスを決定します。

④ ①と同様に再度 Explain 情報を取得し、得た情報のうち Estimated Cost を①と比較します。

<想定例2> 表の断片化によるアクセス効率の低下

【解説】

レコードの追加、更新、削除により以下のような現象が発生して性能劣化につながっている場合があります。

・ページ・オーバーフローが発生して、ディスクI/O効率全般が悪くなる

・あるインデックスの順にデータレコードが配置されていて(あるいは、クラスターインデックスを使用している)効率良いディスクI/O処理ができていたのが、データレコードの配置順が乱れてしまい、ディスクI/O効率が悪くなる。

その状況を改善するためには以下の手順の実施が必要です。

【対応方法】

Runstats -> Rebind -> Reorgchk -> Reorg -> Runstats -> Rebindという順に実行してください。(1 章 4 節データ保守／アプリケーション保守用ツールのご紹介参照)

<想定例3> アプリケーションの自動再バインド発生の場合

【対応方法】

事前にバインド処理を実施しておくことで、アプリケーション実行時の自動再バインドを防ぐ。

【解説】

スキーマ変更などにより(静的)パッケージが無効となった場合、その状態のままアプリケーションを実行しようとすると、自動的に再バインド処理が行われます。この再バインド処理は数秒程度で終わりますが、再バインド処理に伴うシステム・カタログ表へのロックは、実行されたアプリケーションの作業単位の間保持されます。そのため、そのロックと競合するロックを保持しようとするアプリケーションは、このアプリケーションが終了するまで待たされることになります。

5-2-2 特定のコマンドでレスポンスがない場合

5-2-2-1 ロード実行時の場合

以下にこのロード実行時にレスポンスがない場合の障害例とその対応方法を記載いたします。

<想定例1> Load 時の大量な警告発生

【解説】

通常数分で終了するはずのロードが数時間経過しても終了しない場合があります。ロード実行中に於いて db2diag.log にはエラーが出力されない為、外見上処理がハングしているように見えますが、実際には大量な警告メッセージが発生しその処理で時間がかかっていることがあります。この確認は、load query コマンドによって行えます。

db2 “load query table <table-name>”

回復方法は、中断させるか終了するまで待つかのいずれかとなります。中断するにはLOADアプリケーションのFORCE を行いその後再度ロードを行います。中断後、ロード保留状態になった場合、1 章 4-3 節LOADを参照してください。

今後の対策として以下の WARNINGCOUNT オプションも参考にしてください。

＜WARNINGCOUNT オプション＞

ロード処理に於いて警告が出ない事を期待しているが正しいファイルと表が使われているかどうかを検査したい場合、WARNINGCOUNT オプションを使用してオプションの指定個数以上警告が検知されたらロード処理を停止することが出来ます。指定個数がゼロの場合、又はこのオプションが指定されていない場合、何度警告が出されてもロード操作は続行します。警告のしきい値に達した為にロード操作が停止された場合でも、あらためて RESTART モードでロード操作を開始できます。

$ db2 “load from <*file-name*> of <*file-type*> warningcount <*n*> messages <*messagefile-name*> <*load-type*> into <*table-neme*>”

【対応方法】

① ロードを行っているアプリケーションの特定方法

・db2 list applications による特定方法

接続状況が Performing a Load であるか確認してください。

$ db2 list applications show detail

“Status”の列が”Performing a Load”の”Application handle”に注目

・db2 application snapshot による特定方法

接続状況が Performing a Load であるか確認してください。

$ db2 “get snapshot for all application”

Application Snapshot

Application handle = 37

Application status = Performing a Load

② FORCE APPLICATION を行ってロードの停止を実行して下さい。

$ db2 “force application (<*app-handle*>)”

DB20000I FORCE APPLICATION コマンドが正常に終了しました。

DB20124I このコマンドは非同期であり、即時に有効にならない場合もあります。

<db2diag.log 出力結果>

```
2002-06-27-15.20.22.489439   Instance:inst   Node:000
PID:24438(db2agent (SAMPLE))   Appid:*LOCAL.inst.020627060208
database_utilities  DIAG_ERROR   Probe:0   Database:SAMPLE
Error!!! Agent forced -1224, 0,  Detected in file: sqluvld.C, line 2552
2002-06-27-15.20.22.686386   Instance:inst   Node:000
```

```
PID:26286(db2lfrm2)   Appid:*LOCAL.inst.020627060208
database_utilities  sqluldat   Probe:47   Database:SAMPLE
Interrupt received in formatter 2.
2002-06-27-15.20.23.186928   Instance:inst   Node:000
PID:82818(db2lfrm3)   Appid:*LOCAL.inst.020627060208
database_utilities  sqluldat   Probe:47   Database:sample
Interrupt received in formatter 3.
2002-06-27-15.20.23.687138   Instance:inst   Node:000
PID:110922(db2lfrm0)   Appid:*LOCAL.inst.020627060208
database_utilities  sqluldat   Probe:47   Database:SAMPLE
Interrupt received in formatter 0.
2002-06-27-15.20.24.187838   Instance:inst   Node:000
PID:83604(db2lfrm1)   Appid:*LOCAL.inst.020627060208
database_utilities  sqluldat   Probe:47   Database:SAMPLE
Interrupt received in formatter 1.
```

③　ロードのメッセージファイルに出力された SQL エラー内容の解決を行って下さい。

④　②でロードを停止したことによる表スペース状態ロード保留中からの回復(1 章 4-3 節LOAD参照)

$ db2 “load from <*file-name*> of <*file-type*> messages <*messagefile-name*> <*load-type*> into <*table-neme*>”

```
Number of rows read         = 10000
Number of rows skipped      = 0
Number of rows loaded       = 10000
Number of rows rejected      = 0
Number of rows deleted      = 0
Number of rows committed   = 10000
```

※10000 件 LOAD 時の結果

<db2diag.log の出力結果>

```
2002-06-27-15.25.45.395755   Instance:inst   Node:000
PID:167788(db2lrid)   Appid:*LOCAL.inst.020627062544
database_utilities  DIAG_NOTE   Probe:0   Database:SAMPLE
Load CPU parallelism is: 4, 0
2002-06-27-15.25.45.545499   Instance:inst   Node:000
PID:26288(db2lfrm0)   Appid:*LOCAL.inst.020627062544
database_utilities  sqlulPrintPhaseMsg   Probe:0   Database:SAMPLE
Starting LOAD phase at 06/27/2002 15:25:45.439135. Table SAMPLE     .TABLE1
2002-06-27-15.25.45.971437   Instance: inst   Node:000
PID:167788(db2lrid)   Appid:*LOCAL. inst.020627062544
database_utilities  sqlulPrintPhaseMsg   Probe:0   Database:SAMPLE
Completed LOAD phase at 06/27/2002 15:25:45.953597.
```

5-2-2-2 データベースの活動化時(connect または activate 時)にレスポンスがない場合

以下にこのエラーを出力する原因となる障害例とその対応方法を記載いたします。

<想定例1> 破損回復の自動実行処理に時間を要している

【解説】

電源異常などの原因でデータベースが異常終了した後にデータベースを活動化させる際、データベース構成パラメータ AUTORESTART の設定値が”ON”になっていると、自動的に破損回復(故障回復)の処理が実行されます。この破損回復の処理で、ロールフォワードおよびロールバックで適用されるログが大量にあると、データベースが活動化できるまでに長時間を要する場合があります。

【確認方法】

設定の確認方法:

```
$ db2 get db cfg for <database-alias> | grep AUTORESTART
Auto restart enabled                        (AUTORESTART) = ON
```

破損回復の進行状況確認方法:

破損回復開始、終了時には db2diag.log に以下の内容が表示されます。

<db2diag.log 出力例>

```
2002-07-08-11.19.39.800291   Instance:udb7ee   Node:000
PID:59346(db2agent (SAMPLE))   Appid:*LOCAL.udb7ee.020708021938
base_sys_utilities  sqledint   Probe:0   Database:SAMPLE

Crash Recovery is needed.

2002-07-08-11.19.41.314644   Instance:udb7ee   Node:000
PID:59346(db2agent (SAMPLE))   Appid:*LOCAL.udb7ee.020708021938
recovery_manager  sqlpresr   Probe:1   Database:SAMPLE
DIA3908W Crash recovery has been initiated.  Lowtran LSN is "00003E8C800C",
Minbuff LSN is "00003E8C800C".

2002-07-08-11.19.41.402344   Instance:udb7ee   Node:000
PID:59346(db2agent (SAMPLE))   Appid:*LOCAL.udb7ee.020708021938
recovery_manager  sqlprecm   Probe:125   Database:SAMPLE

Using parallel recovery with 5 agents 33 QSets 231 queues and 2 chunks


2002-07-08-11.19.41.667802   Instance:udb7ee   Node:000
PID:59346(db2agent (SAMPLE))   Appid:*LOCAL.udb7ee.020708021938
recovery_manager  sqlprecm   Probe:400   Database:SAMPLE
DIA3916W Forward phase of crash recovery has completed.  Next LSN is
"00003E8C9DC0".

2002-07-08-11.19.41.908510   Instance:udb7ee   Node:000
PID:59346(db2agent (SAMPLE))   Appid:*LOCAL.udb7ee.020708021938
recovery_manager  sqlpresr   Probe:170   Database:SAMPLE
DIA3909W Crash recovery completed.  Next LSN is "00003E8C9DC0".
```

5-2-3 処理中であるが、どの現象にも当てはまらない場合

5-2 節(処理中の可能性が高い場合の対応方法)における内容のいずれにも当てはまらない状態かつ処理中であると判断された場合、以下の対応を行ってください。まず、以下の点を再度チェックして下さい。

【確認手順】

① DB2DIAG.LOGの確認(1 章 2-1-1 節診断ログ(db2diag.log)参照)

```
$ view  <diag_path>/db2diag.log
```

② DB2 の稼働状況の確認

```
$ db2 list applications show detail
```

Lock-Waitのアプリケーションがあった場合、5-3-1 節ロック待ち(Lock-Wait)のプロセスが発生して処理が進まない場合を参照ください。

③ OS 側に CPU モニターツールが存在すれば、それを使用して、大量の CPU 時間を使用しているアプリケーションを判別してください。

上記のチェックにて判断不可能な場合、かつDB2 システムに原因があると思われる場合、8-1 節共通で必要な情報を参照し、サポートセンターに連絡ください。

5-3 ウェイトの可能性が高い場合の対応方法

2-3 節エラーコードは戻らないが、処理のレスポンスがない場合での問題判別で、何らかの処理がされるのを待機していて(CPUは基本的には使用していない)、それ自身の処理は停止している状態と判断された場合、以下の各項のどの現象に該当するかを特定し、対応を行ってください。

5-3-1 ロック待ち(Lock-Wait)のプロセスが発生して処理が進まない場合

【対応】

(ロックを取得しているアプリケーションを特定し、ロックを解放する方法)

① 稼動中のアプリケーションの状況を確認

```
$ db2 list application show detail
```

```
Collected               SAMPLE   /home/taka/taka/NODE0000/SQL00002/
TAKA                            db2bp                28          *LOCAL.taka.011207032039       0
001  1            0                  663570             Lock-wait                                Not
Collected               SAMPLE   /home/taka/taka/NODE0000/SQL00002
TAKA                            db2bp                33          *LOCAL.taka.011207034511       0
001  1            0                  1220752            Lock-wait                                Not
Collected               SAMPLE   /home/taka/taka/NODE0000/SQL00002/
```

出力結果からアプリケーションの 28 と 33 の status が Lock-wait になっていることが判明します。

② ロックのスナップショット採取

LOCKモニター・スイッチをONにしてロック・スナップショットを取得しどのアプリケーションがロックを保留しているかを調べます。ロックのスナップショットで Lock-wait になっているアプリケーションの、ID of agent holding lock 項目を確認して下さい。

```
$ db2 update monitor switches using lock on
```

```
$ db2 get snapshot for locks on <database-alias>
```

```
      Database Lock Snapshot (Snapshot 抜粋)
Application handle                      = 28
Application ID                          = *LOCAL.taka.011207032039
Sequence number                         = 0001
Application name                        = db2bp
Authorization ID                        = TAKA
Application status                      = Lock-wait
Status change time                      = Not Collected
          :                                :
Subsection waiting for lock             = 0
ID of agent holding lock                = 12
Application ID holding lock             = *LOCAL.taka.011207010936
Node lock wait occurred on              = 0
 Lock object type                       = Row
```

スナップショットの結果を見ると、アプリケーション 28 がロックを取得しようとしていたオブジェクトを、アプリケーション 12 がロック取得していたことがわかります。

③ ロックを取得しているアプリケーションの停止

ロックを取得しているアプリケーション ID(<*app_id*>)の特定がなされれば、以下を実施してください。

$ db2 ” f o r c e a p p l i c a t i o n 〈*app_id*〉”

④　問題が解決されたかの確認

①を再度実行し、Lock-Wait のアプリケーションの 28 と 33 の status に変化があることを確認してください。あるいは実行中のアプリケーションよりレスポンスが帰ってくるなどの確認をしてください。

5-3-2 DB2 接続アプリケーションの状態が”UOW EXECUTING”のまま処理が進行しない場合

以下に本節で紹介しておりますこのエラーを出力する原因となる障害例をリストいたします。

○　ログディスクフル設定時の待ち（レジストリーへ変数 DB2_BLOCK_ON_LOG_DISK_FULL）

○　ディスク書込み中断設定による待ちの場合(set write suspend コマンド)

以下にそれぞれの場合の対応方法を記載いたします。

<想定例1> ログディスクフル設定時の待ち(レジストリー変数 DB2_BLOCK_ON_LOG_DISK_FULL)

【解説】

V7.2 より、ログディスクフル時にトランザクション実行を中断（ブロッキング）させることによりトランザクションの失敗を避けることが可能です。ブロッキングされているアプリケーションにはエラーは返らず、UOW_EXECUTING の状態のまま待たされます。DB2 ロガープロセス(db2loggr)は、ログファイルスペースの確保に失敗すると、少なくとも 5 分以内にログファイルスペース確保を再試行し、成功するまで繰り返します。以下にこの障害時の対応方法を記載します。

【対応方法】

①次のコマンドを実行して、DB2 レジストリー変数 DB2_BLOCK_ON_LOG_DISK_FULL の設定値が、”ON”になっていることを確認します。

```
$ db2set DB2_BLOCK_ON_LOG_DISK_FULL
ON
```

・“ON”に設定されていないとき　→　次の<想定例 2>（set write suspend）を参照してください

・“ON”に設定されているとき　→　次の②へ

②db2diag.log に、ログディスクフルのメッセージがあるかを確認する。

＜db2diag.log の例＞

※ このエラーが発生した場合、少なくとも 5 分以内にログの作成が試みれます。ログスペースが確保できるまで更新処理中のアプリケーションは待機させれれます。

※

```
2002-06-13-21.34.13.330449   Instance:db2inst1   Node:000
PID:41144(db2loggr)   Appid:none
data_protection  sqlpgifl   Probe:50
DIA3612C Disk was full.
ZRC=0xFFFFD60C

2002-06-13-21.34.13.483711   Instance:db2inst1   Node:000
PID:41144(db2loggr)   Appid:none
data_protection  sqlpgadf   Probe:550
Disk full on active log path.  DB2 will attempt to create log file again
in 5 minutes.

2002-06-13-21.39.13.748674   Instance:db2inst1   Node:000
PID:41144(db2loggr)   Appid:none
data_protection  sqlpgifl   Probe:50
DIA3612C Disk was full.
ZRC=0xFFFFD60C

2002-06-13-21.39.13.901228   Instance:db2inst1   Node:000
PID:41144(db2loggr)   Appid:none
data_protection  sqlpgadf   Probe:550
Disk full on active log path.  DB2 will attempt to create log file again
in 5 minutes.
```

③十分なログファイル用のスペースがあるかどうかの確認

1.ログファイル用のスペースのパスを確認

$ db2 GET DB CFG FOR <*database-alias*> | grep **パス**

2.ログ用ファイルシステムの空き容量確認

df コマンドによりログ用ファイルシステムの空領域を確認

$ df -k

<実行結果>

```
Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/lvname    1146880       0   100%_    2187   2%   <log_path>
```

※Free:0、%Used:100%になっていないか確認して下さい。

ログファイル用ファイルシステムの空き容量が不足していた場合、以下の対応を行ってください。

④ ログファイル用ファイルシステムの空き領域不足解消

以下のどちらかの手順を実行してください。

1.不要なファイルが存在すれば削除

2.ログファイル用スペースのサイズの拡張

OS のマニュアルを参照してください。

<想定例2> ディスク書込み中断設定による待ちの場合(set write suspend コマンド)

【解説】

V7.2 以降では、データベースのスプリットミラーのようなイメージをオンラインで取得することが可能です。具体的には、set write suspend for database コマンドを発行して表スペースとログに対する物理ディスク書込みを禁止した状態にし、その後ストレージ製品の高速コピー機能を用いてスプリットミラーイメージを取得できます。

SUSPEND 状態にしても、バッファープール内で処理される表データの更新や、ログバッファー内へのログの書込みは許されます。コミット、ログバッファーフル時のログディスク書出し要求や、表スペースのダーティページのディスク書出し要求があると、SUSPEND 状態が解除されて通常の状態に戻るまで、その要求は待機させられます。

SUSPEND 状態では、更新処理だけではなく、参照処理でも物理ディスクへの書込みを伴う場合は中断されることがあることに注意してください。例えば、ORDER BY 句で大量データをソートする SELECT 文の実行で一時表スペースを使用したり、大量データを SELECT することによってバッファープール上のダーティページをディスクに書き出すような処理は中断されます。

注意:この機能と HACMP とをあわせて使用する場合には、V7.2 Fixpak12 以降としてください。

この SUSPEND 状態を解除する方法を以下に記載します。

【対応方法】

①SUSPEND状態を確認してください。

```
$ db2 list tablespaces
```

```
Tablespaces for Current Database
      Tablespace ID                        = 0
      Name                                 = SYSCATSPACE
      Type                                 = System managed space
      Contents                             = Any data
      State                                = 0x10000
      Detailed explanation:
          Write Suspended
```

②データベースの SUSPEND 状態を解除してください。

```
$ db2 set write resume for database
```

5-4 ハングの可能性が高い場合の対応方法

前節までの問題判別で、いずれにも当てはまらない状態(CPU は基本的には使用していない)と判断された場合、以下の対応を行ってください。

まず、本当にハングしているかどうかを確認するため以下の点を再度チェックして下さい。

【確認手順】

① DB2DIAG.LOGの確認(1 章 2-1-1 節診断ログ(db2diag.log)参照)

```
$ view <diag_path>/db2diag.log
```

② DB2 の稼働状況の確認

```
$ db2 list applications show detail
```

Lock-Waitのアプリケーションがあった場合、5-3-1 節ロック待ち(Lock-Wait)のプロセスが発生して処理が進まない場合を参照ください。

③ CPUを使用していないかの確認

```
$ vmstat 2
```

稼働アプリケーションが DB2 システムのみの場合、CPU は使用していないはずです。

④ ディスクにアクセスしていないかの確認

```
$ iostat 2
```

稼働アプリケーションが DB2 システムのみの場合、ディスクにアクセスしていないはずです。

⑤ テンポラリーファイルは作成されていないかの確認(2-3-3-3～2-2-3-5 節参照)

上記のチェックにてハングしていると最終判断した場合、8-2 節ハング時に取得すべき情報を参照し、サポートセンターに連絡ください。

6 エラーコードが表示されない場合の対応方法(アプリケーションに問題がある可能性が高い場合)

アプリケーションによって対応が異なる場合が多いため、ここでは一般的な対応をできる場合に関してのみを記述いたします。アプリケーション用に問題判別マニュアルが存在する場合、まずそちらをご覧になってください。

6-1 アプリケーションに問題がある可能性が高い場合

前節までの問題判別で、DB2 サーバーに接続するアプリケーションに問題があると判断された場合、以下の内容を再度確認してください。

【注意事項】

平常稼働時に以下の事柄はご確認ください。

① 通常どの程度のレスポンスタイムで終了する事を想定してアプリケーションが設計されているか

② アプリケーションそのものの動き(処理フロー)

【確認内容】

①アプリケーション側のログを確認。

②一度も確認していない場合は db2diag.log ファイルを確認。

※詳細は 1章 2-1-1 節診断ログ(db2diag.log)を参照して下さい。

③ダンプファイルを確認。

※詳細は 1章 2-1-3 DB2ダンプ・ファイル/DB2 コアファイルを参照して下さい。

④トラップ・ファイルを確認。

※詳細は 1章 2-1-4 DB2トラップ・ファイルを参照して下さい。

問題解決されない、かつDB2 に原因の可能性がある場合には、8-1 節共通で必要な情報参照の上、必要な情報を収集しサポートセンターと連絡を取って下さい。

7 その他の障害時の対応方法

7-1 ファイル、表スペースの破損、損失時

下記ファイルおよび表スペースが破損あるいは損失された場合、下記のような対応が必要となります。

7-1-1 HISTORY FILE(回復活動記録ファイル)のエラー時(破損、あるいは、損失)

【対応方法】

DB2 の回復活動記録ファイルには正(db2rhist.asc)と副(db2rhist.bak)があります。これらのファイルはデータベースディレクトリーの SQL00001 ディレクトリーに存在します。コマンドが回復活動記録ファイルに対するファイル操作に失敗した場合、SQL2160W や SQL2161N が出力されます。

<記録ファイルの存在状況の確認方法>

適切なパスにて以下のコマンドを実行します。

```
$ ls -l db2rhist*
  -rw-r-----   1 db2inst1      sysadm      4207 Jun 21 11:43 db2rhist.asc
  -rw-r-----   1 db2inst1      sysadm      4207 Jun 21 11:43 db2rhist.bak
```

○ 正、副の一方が破損した場合、db2 list history コマンドは 自動的に 回復活動記録ファイルの修復を試みます。もし、修復のアクションが失敗した場合、 db2diag.log にエラーメッセージが出力されます。db2diag.log に "DB History: Nonrecoverable error - existing history deleted, backup made." の出力がなければ、ファイルは正常 または正常に 修復が完了しています。

<対応例>

```
$ db2 list history backup all for <database-alias>
```

○ 正・副 双方のファイルが "削除" または "不整合な状態" の場合、バックアップ・ファイル内に含まれる回復活動記録ファイルを戻して回復します。ただし、戻したバックアップ以降の記録は回復できません。

<対応例>

```
$ db2 restore database <database-alias> history file from <backupimage-path> taken at
<time-stamp>
```

7-1-2 一時表スペースの破損

【解説】

REGULAR、LONG、TEMPORARY の表スペースが壊れてもその表スペースのステータスが OFFLINE になり、データベースへの接続は可能です。破損が一時表スペースならば、データベースへの接続後、新規の一時表スペースを作成し、壊れた一時表スペースを削除することで回復できます。または、何らかの操作でその表スペースがアクセス可能な状態に回復することができた場合には、ALTER TABLESPACE ... SWITCH ONLINE ステートメントを実行することでも回復可能です。これは、物理的なディスク障害の場合でも、ファイルシステムがマウントされていなかった場合でも同様です。

また循環ログ方式の場合に限り、データベース接続時に破損回復が必要で、且つ一時表スペースに

障害が発生している場合には、まずRESTARTコマンドにより、DROP　PENDING TABLESPACEオプションを指定して実行し正常終了させることが必要です。(1 章 1-4-4-4 節一時表スペースが壊れた場合の回復参照)

【対応方法 1】

以下に、新規の一時表スペースを作成し、壊れた一時表スペースを削除する方法を記述いたします。

<障害時の db2diag.log 出力例>

```
2002-06-26-15.37.28.444075   Instance:db2inst1   Node:000
PID:113568(db2agent (sample))   Appid:*LOCAL. smith.020626063727
buffer_pool_services  sqlbStartPools   Probe:30   Database:sample
Tablespace TEMPSPACE1 (3) is in state x0.
Starting it failed. rc=ffff c11e

2002-06-21-15.37.28.776916   Instance:db2inst1   Node:000
PID:113568(db2agent (sample))   Appid:*LOCAL. smith.020626063727
buffer_pool_services  sqlbStartPoolsErrorHandling   Probe:38   Database:sample

        Tablespace TEMPSPACE1 is set to be OFFLINE
```

① データベースに接続

$ db2 connect to <*database-alias*>

※ データベースに接続できない場合は 1 章 1-4-4-4 節循環ログ方式で一時表スペースが壊れた場合の回復を参照してください。

② 表スペースの状況を表示

```
$ db2 list tablespaces
 Tablespace ID                        = 3
  Name                                 = TEMPSPACE1
  Type                                 = System managed space
  Contents                             = System Temporary data
  State                                = 0x4000
    Detailed explanation:
          Offline
```

③ 新規の一時表スペースを作成

$ db2 “create temporary tablespace <*tablespace-name*> managed by system using (‘<*container string*>’)”

```
      DB20000I  ＳＱＬ　コマンドが正常に終了しました。
```

④ 表スペースの状況を再度表示し、確認

```
$ db2 list tablespaces
       Tablespace ID                        = 3
        Name                                 = TEMPSPACE1
        Type                                 = System managed space
        Contents                             = System Temporary data
        State                                = 0x0000
          Detailed explanation:
            Normal
```

⑤ 壊れた一時表スペースを削除

$ db2 “drop tablespace <*tablespace-name*>”

DB20000I　ＳＱＬ　コマンドが正常に終了しました。

【対応方法 2】

以下に、マウントされていない等の原因を解決でき表スペースを使用可能状態にできる場合の対応方法を記述いたします。

① データベースに接続

$ db2 connect to <*database-alias*>

② 表スペースの状況を表示

$ db2 list tablespaces

```
 Tablespace ID                        = 3
  Name                                 = TEMPSPACE1
  Type                                 = System managed space
  Contents                             = System Temporary data
  State                                = 0x4000
    Detailed explanation:
          Offline
```

③ OS 等の操作により表スペースを使用可能な状態に改善

④ ALTER TABLESPACE ... SWITCH ONLINE ステートメントを実行

$ db2 “alter tablespace <*tablespace-name* > swich online”

<コマンド実行時の db2diag.log 出力例>

```
        2002-06-21-15.39.24.067969   Instance:db2inst1   Node:000
PID:113568(db2agent (sample))   Appid:*LOCAL.smith.020626063727
buffer_pool_services  sqlbSwitchTablespaceOnline   Probe:883   Database:sample

Tablespace TEMPSPACE1 (3) has been brought online.  New state = 0x00000000.
```

あるいは、全てのデータベース接続を切断してデータベースの接続しなおし

$ db2 force application all

$ db2 terminate

$ db2 connect to <*database-alias*>

⑤ 表スペースの状況を再度表示し、確認

$ db2 list tablespaces

```
       Tablespace ID                        = 3
        Name                                 = TEMPSPACE1
        Type                                 = System managed space
        Contents                             = System Temporary data
        State                                = 0x0000
          Detailed explanation:
            Normal
```

7-1-3 ユーザー表スペースの破損

【解説】

一時表スペース同様ユーザー表スペースが壊れてもその表スペースのステータスがOFFLINEになり、データベースへの接続は可能です。

一般的な回復方法は、バックアップ・ファイルから復元を行うことです。または、何らかの操作でその表

スペースがアクセス可能な状態に回復することができた場合には、ALTER TABLESPACE ... SWITCH ONLINE ステートメントを実行することでも回復可能です。これは、物理的なディスク障害の場合でも、ファイルシステムがマウントされていなかった場合でも同様です。
また循環ログ方式の場合に限り、データベース接続時に破損回復が必要で、且つ表スペースに障害が発生している場合には、まずRESTARTコマンドにより、DROP PENDING TABLESPACEオプションを指定して実行し正常終了させることが必要です。(1 章 1-4-4-4 節一時表スペースが壊れた場合の回復参照)

【対応方法】

<障害時の db2diag.log 出力例>

```
2002-06-21-18.47.17.940099   Instance:db2inst1   Node:000
PID:64566(db2agent (TESTDB))   Appid:*LOCAL.db2inst1.020621094717
buffer_pool_services  sqlbStartPools   Probe:30   Database:TESTDB

Tablespace USERSPACE1 (2) is in state x0.
Starting it failed. rc=ffff c11e                                    ..ｦ.

2002-06-21-18.47.18.072964   Instance:db2inst1   Node:000
PID:64566(db2agent (TESTDB))   Appid:*LOCAL.db2inst1.020621094717
buffer_pool_services  sqlbStartPoolsErrorHandling   Probe:38   Database:TESTDB

Tablespace USERSPACE1 is set to be OFFLINE
```

① 表スペースの状況を表示

```
$ db2 list tablespaces
   Tablespace ID                        = 2
    Name                                 = USERSPACE1
    Type                                 = System managed space
    Contents                             = Any data
    State                                = 0x4000
      Detailed explanation:
            Offline
```

② db2 list history backup コマンドで最新のバックアップとログを確認します。

```
$ db2 list history backup all for <database-alias>
```

③ ②で確認したバックアップ・パスにバックアップ・ファイルがあることを確認して下さい。

```
$ ls <baskup-path>
  SAMPLE.0.inst.  NODE0000.CATN0000.  20020621181335.001
```

④ ②で確認した"Earliest Log"以降がログ・パスかアーカイブ・パス(USEREXIT 使用時)にあることを確認して下さい。

```
$ db2 get database cfg for <database-alias> | grep パス
ログファイルのパス                  = <log_path>
$ ls <log_path>
  S0000005.LOG S0000006.LOG
```

⑤ バックアップ取得以降にロード処理が実行されているか確認して下さい。 ロード処理がなければ ⑦に進んで下さい。ロード処理がありロードをプション copy yes にて実行されている場合(1 章 4-3 節LOAD参照)、データ・コピー・ファイルがコピー・ファイル・パスにあるか確認して下さ

い。

・バックアップ以降にロードが実行されているかの確認

（ロードが実行されていない場合）

```
$ db2 list history load since <time-stamp> for <database-alias>

      List History File for sample

    Number of matching file entries = 0
```

（ロードが実行されている場合）

```
$ db2 list history load since <time-stamp> for <database-alias>

      List History File for sample
    Number of matching file entries = 1
  Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log  Backup ID
   --  ---  ------------------  ----  ---  ------------  ------------
-------------------------- ------------
   C T 20020621112639002 R S S0000005.LOG S000000６.LOG 20020627162432001

------------------------------------------------------------------------------------------
--- --------
    "INST     "."TABLE1" resides in 1 tablespace(s):
    00001 USERSPACE1

------------------------------------------------------------------------------------------
------------
     Comment: DB2
    Start Time: 20020621112639
    End Time: 20020621122616

------------------------------------------------------------------------------------------
------------
    00002
Location: <location-path>
```

上記結果の<*location-path*>もとに以下を実行しコピーファイルの存在を確認して下さい。

```
$ ls <location-path>
```

⑥ ②で採取したバックアップファイルからリストア後、ロールフォワード回復を行って下さい。

$ db2 “restore database <*database-alias*> tablespace <*tablespace-name*> from <*backupfile-directory*> taken at <*time-stamp*>”

DB20000I RESTORE DATABASE コマンドが正常に終了しました。

```
$ db2 “rollforward db <database-alias> to end of logs and stop tablespace <tablespace-name>”

                    Rollforward Status
    Input database alias                = sample
    Number of nodes have returned status  = 1
    Node number                       = 0
    Rollforward status                  = not pending
    Next log file to be read              = S0000005.LOG
    Log files processed                 =  -
        Last committed transaction          = 2002-06-21-09.13.56.000000
```

DB20000I ROLLFORWARD コマンドが正常に終了しました。

8 サポートセンター連絡時に取得すべき情報

対応が困難な障害の場合、サポートセンターに御連絡いただくことになりますが、その際、本節の操作を実行しあらかじめ情報を収集していただくと役立ちます。これらの情報を2章1節障害時の履歴および環境の保存にて収集した情報とあわせサポートセンターに送付ください。

8-1 共通で必要な情報

8-1-1 V7.1FP4 以降(db2support コマンドが提供されている)の場合

① 調査対象となるインスタンスが起動していることを確認して下さい。

$ db2_ps

(db2syscなどのプロセスが出力される事を確認して下さい。インスタンスが起動していなければ db2startで起動して下さい。)

② DB2コアファイルまたはシステムコアファイル(core)が存在している場合、付録Aのシェルを以下の手順で実行してください。

<使用方法>

5. コアファイルの存在を確認します。(1章 2-1-3節DB2コアファイル、2-3-3節core参照)
6. coreが発生するとエラーログにCORE_DUMPのエラーラベル名で記録されるので、その記録より実行プログラム名を判別してください。
7. 2.で判別したプログラム名より、whichコマンドなどを使用し、実行可能ファイルのフルパスを取得して下さい。

   <使用例>

   ```
   $ which db2sysc
     $INSTHOME/sqllib/adm/db2sysc
   ```
8. データベース・マネージャー構成パラメーターのDIAGPATH上で、dbxtraceという名称にて付録Aの内容のシェルを作成してください。

   ```
   $ vi dbxtrace
   $ chmod u+x dbxtrace
   ```
9. 実行環境を必ず英語にしてください。

   ```
   $ export LANG=C
   ```
10. コア・ディレクトリーごとに以下のコマンドを実行してください。(事前にテスト環境にて稼働を確認してください。)

    /dbxtrace <*executable*> ./c<*PID*>.000/core > dbxtrace.log 2>&1

    ※ <*executable*>:3で取得した実行可能ファイルのフルパス

③ ある程度容量のあるパスに移動し、インスタンスオーナーで下記を実行して下さい。

$ db2support . -d <*database-alias*> -c -n <*PMR#*> > db2support.log 2>&1

<使用例>

$ db2support . -d sample -c -n 11223.b000.c760 > db2support.log 2>&1

PMR#は一つ一つの障害事象に割り当てられる管理番号で、通常サポートセンターにご連絡いた

だいた際に入手できます。コマンド実行後、db2support.zip及びdb2support.logという2つのファイルができますので、両方送付して下さい。
また、上記の例のように実際に実行されたdb2supportコマンドも併せてお知らせ下さい。

④ db2supportコマンドにより自動的にdb2diag.logも収集されますが、ファイルが途中で途切れてしまい問題発生時の出力が得られない事があります。 誠にお手数ですが、上記db2supportの出力に加えてdb2diag.logもあわせて別途送付頂きますようお願いいたします。

8-1-2 V7.1FP4 以前(db2support が提供されていない)の場合 (参考)

※以下の情報は「LANG＝En_US」にしてから実行して下さい。

① syslog（設定していない場合は不要）

② DB2コアファイルまたは、システムコアファイル(core)が存在している場合、付録Aのシェルを以下の手順で実行してください。

<使用方法>

1. コアファイルの存在を確認します。(1章 2-1-3節DB2コアファイル、2-3-3節core参照)
2. coreが発生するとエラーログにCORE_DUMPのエラーラベル名で記録されるので、その記録より実行プログラム名を判別してください。
3. 2.で判別したプログラム名より、whichコマンドなどを使用し、実行可能ファイルのフルパスを取得して下さい。

   <使用例>

   ```
   $ which db2sysc
     $INSTHOME/sqllib/adm/db2sysc
   ```

4. データベース・マネージャー構成パラメーターのDIAGPATH上で、dbxtraceという名称にて付録Aの内容のシェルを作成してください。

   ```
   $ vi dbxtrace
   $ chmod u+x dbxtrace
   ```

5. 実行環境を必ず英語にしてください。

   ```
   $ export LANG=C
   ```

6. コア・ディレクトリーごとに以下のコマンドを実行してください。(事前にテスト環境にて稼働を確認してください。)

   /dbxtrace <*executable*> ./c< *pid* >.000/core > dbxtrace.log 2>&1

   ※ <*executable*>:3で取得した実行可能ファイルのフルパス

③ db2diag. logを含むデータベース・マネージャー構成パラメーターのDIAGPATHに障害発生日に作成された全てのファイル

（デフォルトでは、$INSTHOME/sqllib/db2dumpパス以下）

④ データベース構成パラメーターの設定内容

```
$ db2 get db cfg for <database-alias>
```

⑤ データベース・マネージャー構成パラメーターの設定内容

```
$ db2 get dbm cfg
```

⑥ 修正パック情報(以下の各コマンドの出力)

```
$ lslpp -ah
$ db2level
$ dump -H $INSTHOME/sqllib/lib/libdb2e.a
$ dump -H $INSTHOME/sqllib/lib/libdb2.a
```

(AIX 64bit版をお使いのお客様の場合)

```
$ dump -H -X64 $INSTHOME/sqllib/lib/libdb2e.a
$ dump -H -X64 $INSTHOME/sqllib/lib/libdb2.a
```

⑦ エラーログ (以下のコマンドの出力)

```
$ errpt -a
```

⑧ H／W、S／W情報 (以下の各コマンドの出力)

```
$ lscfg -v
$ ulimit -a(インスタンスオーナーで実行して下さい)
```

⑨ インスタンスプロファイル

```
$INSTHOME/.profile
$INSTHOME/sqllib/db2profile
```

⑩ レジストリー変数の設定(以下のコマンドの出力)

```
$ db2set -all
```

⑪ データベースディレクトリー情報(以下のコマンドの出力)

```
$ db2 list db directory
```

⑫ ノードディレクトリー情報(以下のコマンドの出力)

```
$ db2 list node directory
```

※以下のコマンドは「LANG=Ja_JP」等、データベースのコードページにて CONNECT してから実行して下さい。

⑬ 表スペース情報

```
$ db2 list tablespaces show detail
```

8-2 ハング時に取得すべき情報

【資料採取時の注意事項】

・ハングしている状態で資料を収集する必要があります。

・db2_kill, システムのリブート等でハング状態を解消した後では有効な資料は取得できません。

【資料採取手順】

まずはじめに、本当にハングしているかどうかを確認するため、 以下の点を再度チェックして下さい。

5-4 節ハングの可能性が高い場合の対応方法ですでに実行済みの場合は不要です。

【確認手順】

① DB2DIAG.LOG の確認

```
$ view <diag_path>/db2diag.log
```

② DB2 の稼働状況の確認

```
$ db2 list applications show detail
```

Lock-Waitのアプリケーションがあった場合、5-3-1 節ロック待ち(Lock-Wait)のプロセスが発生して処理が進まない場合を参照ください。

③ CPUを使用していないかの確認

```
$ vmstat 2
```

稼働アプリケーションが DB2 システムのみの場合、CPU は使用していないはずです。

④ ディスクにアクセスしていないかの確認

```
$ iostat 2
```

稼働アプリケーションが DB2 システムのみの場合、ディスクにアクセスしていないはずです。

⑤ テンポラリーファイルは作成されていないかの確認(2-3-3-3～2-2-3-5 節参照)

上記のチェックにてハングしていると判断した場合は、注意事項を必ず確認後資料を取得して下さい。

＜注意事項1＞

db2 force application all コマンドで全アプリケーションを強制終了させ、ハングしているアプリケーションだけを残します。

注:上記コマンドを入力すると全アプリケーションを強制終了させますので、使用の際は十分注意して使用して下さい。

＜注意事項2＞

db2のコマンドが効かない(プロンプトが返ってこない)場合も必ず、資料採取手順のうち以下は実行ください。

```
$ db2trc
$ ipcs -a
$ lsps -a
$ vmstat
$ iostat
```

注意事項をお読みいただいた後、下記手順で資料採取を行ってくだい。

1 モニタースイッチをオン。

```
$ db2 update monitor switches using bufferpool on lock on sort on statement on table on uow on
```

2 kill -36

```
$ kill -36 < pid >
```

(< *pid* >:注意事項1で入力した$db2 force application all で残ったプロセスの ID をセット)

3 snapshot

```
$ (date;db2 get snapshot for all on <dastabaes-alias>)>snapshot.txt
```

4 list applications (活動中のデータベース・アプリケーションの関する情報を出力します)

```
$ (date;db2 list applications show detail) >listapp.txt
```

5 ps (現在のプロセスの状態を出力します)

```
$ (date;ps -aef) > psaef.txt
```

```
$ (date;ps aux) > psaux.txt
```

6 ipcs (アクティブなプロセス間通信機能についての情報を出力します)

```
$ (date;ipcs -a) > ipcs.txt
```

7 2分待つ

8 kill -36(2回目)

```
$ kill -36 < pid >
```

9 snapshot(2回目)

```
$ (date;db2 get snapshot for all on <database-alias>)>>snapshot.txt
```

10 list applications(2回目)

```
$ (date;db2 list applications show detail)>>listapp.txt
```

11 ps(2回目)

```
$ (date;ps -aef)>>psaef.txt
$ (date;ps aux) >> psaux.txt
```

12 ipcs(2回目)

```
$ (date;ipcs -a) >> ipcs.txt
```

13 2分待つ

14 kill -36(3回目)

```
$ kill -36 < pid >
```

15 snapshot(3回目)

```
$ (date;db2 get snapshot for all on <database-alias>)>>snapshot.txt
```

16 list applications(3回目)

```
$ (date;db2 list applications show detail)>>listapp.txt
```

17 ps(3回目)

```
$ (date;ps -aef)>>psaef.txt
$ (date;ps aux) > psaux.txt
```

18 ipcs(3回目)

```
$ (date;ipcs -a)>> ipcs.txt
```

19 モニタースイッチをオフ

```
$ db2 update monitor switches using bufferpool off lock off sort off st atement off table
off uow  off
```

20 CPU/メモリーを一番保有している db2 process に対して db2trc

```
$ db2trc on -p < pid > -l 1000000 -e -1
```

21 1分以上待ち、以下を実行

```
$ kill -60 < pid >
$ db2trc dmp trace.< pid >.dmp
$ db2trc off
$ db2trc fmt trace.< pid >.dmp trace.fmt
$ db2trc flw trace.< pid >.dmp trace.flw
```

22 db2trc の取得

```
$ db2trc on -l 10000000 -e -1
```

23 1 分以上待ち、以下を実行

```
$ db2trc dmp db2trc.dmp
$ db2trc off
$ db2trc flw db2trc.dmp db2trc.flw
$ db2trc fmt db2trc.dmp db2trc.fmt
```

以上の手順で採取した資料とあわせて 8-1 節共通情報採取で調査に必要な資料を取得する。

3章　参考資料・文献

当ガイドの作成にあたり、以下の各種資料を参照しています。

1 DB2 マニュアル

- 「問題判別の手引き」
- 「管理の手引き」
- 「SQL 解説書」
- 「SQL 概説およびインストール」
- 「コマンド解説書」
- 「メッセージ解説書」
- 「システム・モニター　手引きおよび解説書」
- 「DB2コネクト使用者の手引き」
- 「CLI ガイドおよび解説書」

上記資料はオンラインマニュアルとしても以下のサイトより入手可能です。

- V8 インフォメーション・センター　日本語版、英語版（HTML 形式のみ）
  http://www.ibm.com/jp/software/data/developer/library/manual/db2online/index.htm
- V8 日本語版　（PDF 形式）
  http://www.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/v8pubs_trans.d2w/report
- V8 英語版　　（PDF 形式）
  http://www.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/v8pubs.d2w/en_main
- V8 内部エラーコード　　（PDF 形式）
  http://www-1.ibm.com/support/docview.wss?rs=71&context=SSEPGG&q=0x0258000C&uid=swg21066178&loc=en_US&cs=utf-8&lang=en+en

マニュアルの最新版は日本語訳されていないものがあります。
その際は、英語マニュアルのサイトをチェックください。

2 AIX 関連

- 「AIX 5L V5.2 ドキュメンテーション」
  http://publib16.boulder.ibm.com/cgi-bin/ds_form?lang=Ja_JP&viewset=AIX
  本書には、日常の業務においてシステム管理者として実行する主なタスク、および AIX がシステム管理用に提供するツールに関する情報が収められています
- 「DB2 UDB on AIX 4.3.3, 5.1, 5.2 の既知の問題について」（英語）
  http://www-1.ibm.com/support/docview.wss?rs=71&context=SSEPGG&q1=IY49385&uid=swg21165448&loc=en_US&cs=utf-8&lang=en+en。

3 一般書籍

- 「DB2 ユニバーサル・データベース オフィシャルガイド」
  ジョージ・バラクーズ、ビル・ウォン著

日本アイ・ビー・エム(株) ナショナル・ランゲージ・サポート訳
菅原香代子監訳
ピアソン・エデュケーション

- 「DB2 技術全書」
  日本アイ・ビー・エム システムズ・エンジニアリング㈱
  日本アイ・ビー・エム㈱ 編・著
  アスキー出版

4 その他の参考情報(日本語版 Web サイト)

- 「初めての DB2 UDB 障害特定・対策チェックリスト」
  http://www.ibm.com/jp/software/data/developer/pdchecklist/
- 「問題判別習熟ガイド」
  http://www-6.ibm.com/jp/software/data/developer/pd/
- 「DB2 Developer Domain」
  http://www.ibm.com/jp/software/data/developer/
  Web サイトでは、新着情報・技術白書・最新記事・テクニカル情報・スキル育成・交換情報に関する情報が掲載してあります。
- 「DB2 フォーラム」
  http://wsp01.alpha-mail.ne.jp/FRM/DB2
  RDB 技術者の情報交換ネットワークサイトです。
- 「DB2 ホームページ」
  http://wwwibm.com/jp/software/data/
  最新ニュース・製品・ソリューション情報を掲載しています。
- 「DB2 University Library」
  http://www.ibm.com/jp/software/data/db2univ/library
  DB2 に関する技術情報を掲載しています。
- 「ソフトウェア・サポート」
  http://www.ibm.com/jp/software/support/
  エラー回避方法や不具合修正方法等をお届けする技術速報、およびWeb経由での技術的な問題の報告や照会を行うことができます。
  従来当ページで提供していた技術情報検索機能は2004年7月より以下のURLに移動となります。
- 「技術情報検索」(2004 年 7 月以降)
  http://www.ibm.com/jp/software/tech/search/
  ソフトウェアに関する技術情報がキーワード検索できます。
- 「ソフトウェア技術情報」(2004 年 7 月以降)
  http://www.ibm.com/jp/software/tech/
  目的別に役に立つ技術情報リンクをまとめたポータルです。

5 その他の参考情報（英語版 Web サイト）

- 「DB2 World Wide Web」
  http://www.ibm.com/software/data/
  ページには、ユース、製品説明、研修スケジュールなどの DB2 に関する最新情報が提供さています。ただし、提供されている情報は英語です。
- 「DB2 Product and Service Technical Library」
  http://www.ibm.com/software/data/db2/library/
  よくされる質問（FAQ）、修正内容、資料、および最新の DB2 技術情報などが情報へのアクセスが提供されています。
- 「DB2 UDB サポート（US）」
  http://www.ibm.com/software/data/db2/udb/support.html

  上記のリンク先 URL は、2004 年 6 月 23 日現在のものであり、予告なく変更される場合があります。

付録

付録 A

本節では AIX サーバー上の実システムを想定し、障害時にサポートセンターに送付すべき情報を取得するためのシェルスクリプトの例を紹介いたします。(シェルスクリプトは K シェル用です。)
本節のスクリプトを実環境に適用する場合は、必ず事前に正常動作することを確認してください。

```
#!/bin/ksh

# Automate getting the dbx trace information
# ------------------------------------------------------
# AIX V4 32bit max thread limit in sys/thread.h
# Customer can adjust this value to max of target process
# ------------------------------------------------------
integer MAXTHREADS=512
integer MINTHREADS=1
integer THREAD_NUM=$MINTHREADS

#
# Usage
#

PROGNAME=$(basename $0)
function Usage {
        print "$(basename $0): Automate getting dbx trace information " 1>&2
        print "" 1>&2
        print "For core files:" 1>&2
  print "    Usage:   $PROGNAME [executable] [core]" 1>&2
  print "      or :   $PROGNAME -c corefile" 1>&2
  print "    Example: $PROGNAME /usr/jdk_base/bin/aix/native_threads/java core"  1>&2
        print "    (Please make sure you use the java executable and not the java script)" 1>&2
        print "" 1>&2
        print "To attach to a running or hung process" 1>&2
  print "    Usage:   $PROGNAME -a PID"  1>&2
  print "    Example: $PROGNAME -a 1234"  1>&2
  exit -1
}

#
# Parse command lime arguments
#
typeset PID
EXECUTABLE=/usr/jdk_base/bin/aix/native_threads/java
CORE_FILE=core

if [[ $# > 0 ]]
then
    if [[ $1 = "-a" ]]
    then
       if [[ $# -eq 2 ]]
       then
           PID=$2
```

```
       else
           Usage
       fi
    elif [[ $1 = "-c" ]]
    then
       if [[ $# -eq 2 ]]
       then
           CORE_FILE=$2
       else
           Usage
       fi
    else
        EXECUTABLE=$1
        if [[ $# -eq 2 ]]
        then
            CORE_FILE=$2
        fi
    fi
fi

#
# Check command lime arguments
#     then set the appropriate dbx commands
#
if [[ $PID. -eq . ]]
then
    if [[ ! ( -f $EXECUTABLE ) ]]
    then
  print "executable file: $EXECUTABLE does not exist" 1>&2
  Usage
    fi

    if [[ ! ( -x $EXECUTABLE ) ]]
    then
  print "executable file: $EXECUTABLE is not executable" 1>&2
  Usage
    fi

    if [[ ! ( -f $CORE_FILE ) ]]
    then
  print "core file: $CORE_FILE does not exist" 1>&2
  Usage
    fi
    if [[ $CORE_FILE = core ]]
    then
        print "****************************************" 1>&2
        print "* Failure of this script or dbx may     *" 1>&2
        print "* overwrite your existing core file.    *" 1>&2
        print "* It is recomended that you rename your *" 1>&2
        print "* existing core file and use the -c flag *" 1>&2
        read Y_OR_N?"* Do you wish to continue (y/n): " 1>&2
        print "****************************************" 1>&2
        if [[ $Y_OR_N. != y. ]] && [[ $Y_OR_N. != Y. ]]
        then
            exit 0
        fi
    fi
    typeset -r DBX_COMMAND="/usr/bin/dbx $EXECUTABLE $CORE_FILE"
```

```
    typeset -r DETACH="quit"
else
    ps $PID > /dev/null 2>&1
    if [[ $? != "0" ]]
    then
        print "Process ID : $PID does not exist" 1>&2
 Usage
    fi
    typeset -r DBX_COMMAND="/usr/bin/dbx -a $PID"
    typeset -r DETACH="detach"
fi


typeset THREAD_SUB_COMMAND
typeset -r WHERE="where"
typeset -r REGISTERS="registers"
typeset -r MAP="map"
typeset -r DIVIDER="------------------------------------------------------------------------"
typeset -r DIVIDER2="===================="

# Create thread sub command statements
print "Creating subcommand file...." 1>&2
typeset -r COMMAND_FILE=./dbx_commands.$$
awk -v thread_num=$THREAD_NUM -v max_threads=$MAXTHREADS -vdivider=$DIVIDER -vdivider2=$DIVIDER2
'
  BEGIN {
    print "thread"
    print "print ¥"¥""
    print "thread info"
    print "print ¥"" divider "¥""
    while (thread_num <= max_threads) {
      print "thread " thread_num
      print "thread current " thread_num
      print "print ¥"" divider2 "¥""
      print "where"
      print "print ¥"" divider2 "¥""
      print "registers"
      print "print ¥"" divider "¥""
      thread_num++
    }
    print "map"
    print "print ¥"¥""
    print "mutex"
    print "print ¥"¥""
    print "condition"
    print "print ¥"" divider "¥""
  }
' < /dev/null >> $COMMAND_FILE
echo "$DETACH" >> $COMMAND_FILE

# Remove the tail end information about non existant threads
typeset -r REMOVE_MISSING_THREADS="/^['][¥$]t[0-9]*['] is not an existing thread./,/^$DIVIDER/d"
typeset -r REMOVE_BAD_LINES="/^program is not active¥$/d"

print "Running dbx..." 1>&2
$DBX_COMMAND < $COMMAND_FILE 2>&1 | sed -e "$REMOVE_MISSING_THREADS" -e "$REMOVE_BAD_LINES"

rm $COMMAND_FILE
```

```
if [[ $? != "0" ]]
then
    print "failed to remove $COMMAND_FILE" 1>&2
fi
print "dbx has ended with RC=$?" 1>&2
```
