

WASV8.0によるWebシステム基盤設計Workshop

Designing Web System Infrastructure with WAS V8.0

データベース接続設計

## 当セッションの目的

- WASとデータベースとの接続に必要な知識の習得
  - ◆ WASとデータベースを使用したシステムのトポロジー設計を理解する。
  - ◆ WASとDB2との接続におけるJDBCドライバー設計、データ・ソース設計、アプリケーション設計、パフォーマンス/問題判別手法を理解する。
  - ◆ WASとOracle RAC、WASとOracle FCFとの接続における設計手法を理解する。

## Agenda

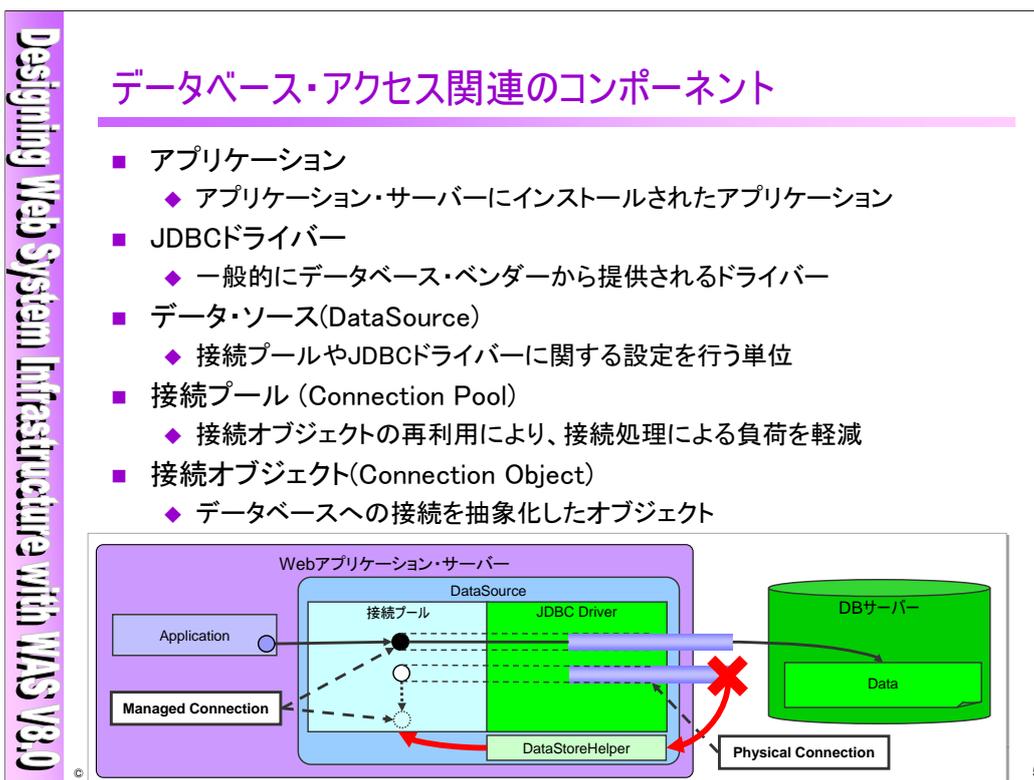
---

1. トポロジー
2. WAS-DB2接続設計
  - 2-1 JDBCドライバー設計
  - 2-2 データソース設計
  - 2-3 アプリケーション設計
  - 2-4 パフォーマンス / 問題判別
  - 2-5 Hints & Tips
3. WAS-Oracle接続設計
  - 3-1 WAS-Oracle RAC接続設計
  - 3-2 WAS-Oracle FCF接続設計

まとめ・参考文献

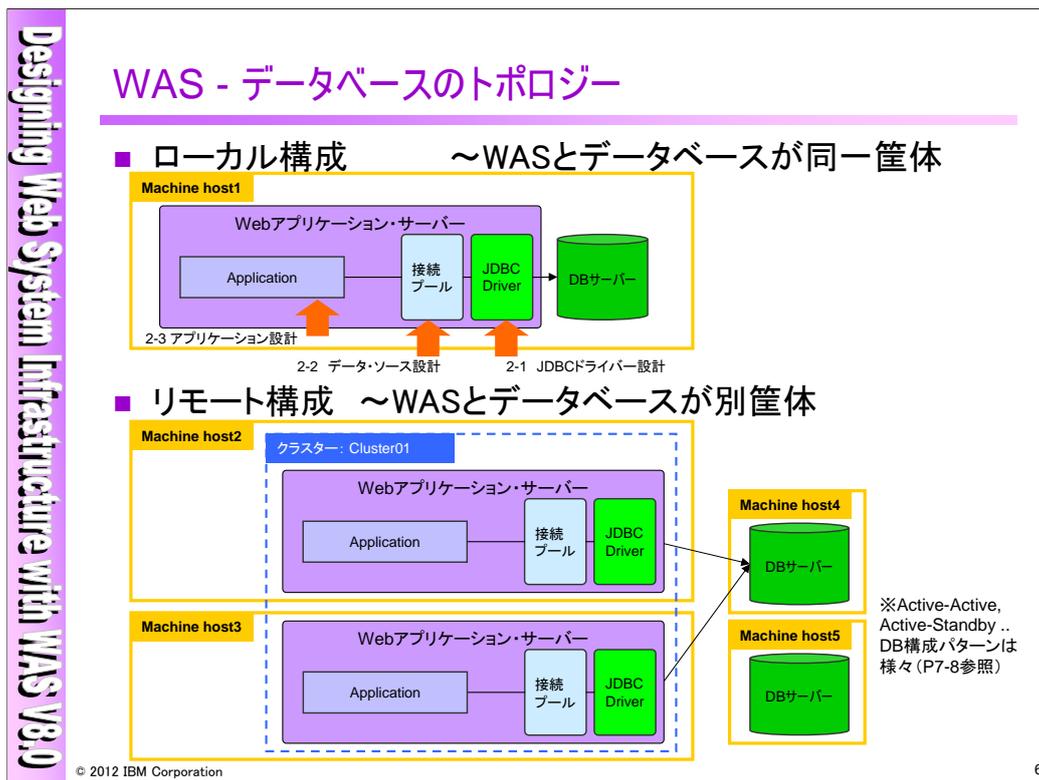


1章では、WebSphere Application Server(以降、WAS)からデータベースにアクセスする際に必要となるコンポーネントと、WASとデータベースを使用したシステムのトポロジーについてご説明致します。



データベースにアクセスする際に必要となる主なコンポーネントは、アプリケーション、データ・ソース、接続プール、接続オブジェクト、JDBCドライバーです。

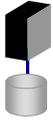
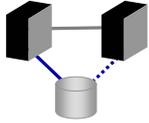
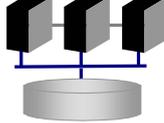
Webアプリケーション・サーバーとしてWASを使用した場合、WASからデータベースへアクセスする際には、WAS上にJDBCドライバー、データ・ソースを定義します。さらに、データ・ソースで接続プールなどを設定します。接続プールの設定は、実際の物理接続(Physical Connection)ではなく、Managed Connectionに対して行います。DataStoreHelperは、SQLCodeとSQLStateから例外の種類を判別します。



WASとデータベースを使用したシステムの代表的なトポロジーは、ローカル構成とリモート構成に分けることができます。

ローカル構成は、最もシンプルな構成であり、WASとデータベースを同一筐体に配置します。JVM、データベースがそれぞれ一つずつしか存在しないため、高可用性や高拡張性は実現できません。従いまして、これらの要件が必要でないテスト環境や開発環境などで採用します。

リモート構成では、WASとデータベースを別筐体に配置します。WASサーバーはWAS ND構成のクラスタリング機能、DBサーバーはPowerHA等のクラスタリングソフトウェアを使用することにより、高可用性と高拡張性を実現します。主に本番環境などで採用します。

データベースのトポロジー(1/2)			
アーキテクチャー	スタンド・アロン構成	コールド・スタンバイ構成 (HA構成)	ロード・バランス型 クラスタリング構成
			
DBのアーキテクチャー	スタンド・アロン	アクティブ・スタンバイ型	アクティブ・アクティブ型 シェアード・ディスク型 製品例: DB2 pureScale, Oracle RAC
概要	DBサーバー1台構成	一時点で稼動するDBサーバーは1台となる構成。サーバー機器障害に備えた冗長化構成	構成サーバー全てでDBサーバー・プロセスが稼動し、クライアントからの要求を処理する
WASからのDB接続定義	DBサーバーのホスト名 (IP)、Portは常に同一	DBサーバーのホスト名 (IP)、Portは常に同一になる構成とする	接続先となるDBサーバー全てのホスト名 (IP)、Port をWASデータソースに登録しておく構成が一般的
DB障害時のサービス影響	DBサーバーの復旧までサービスが完全停止	IP/ディスクの引き継ぎが完了し、スタンバイ側のDBが起動するまでサービス停止となる	全サーバーに障害が起きない限りDB接続は可能。但し単一サーバー障害時は、ログからの回復が完了するまで該当データへのアクセスが制限される。

© 2012 IBM Corporation

7

先ほどのページでは、WASとデータベース・サーバーが同居するか否かの観点のトポロジーをご紹介しました。

こちらの2ページでは、データベース・サーバーのトポロジーについて代表的なものをご紹介します。

最もシンプルなのはスタンドアロン構成です。

DBサーバー1台のみで構成されているため、データベース・サーバーに障害が発生すると、DBデータを利用する業務アプリケーションは利用できなくなります。データベースがSPOFになっている状態といえます。

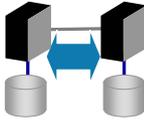
このようなSPOFの状態を解消するため、最もスタンダードに取られる構成がアクティブ・スタンバイ型の構成です。HA構成、コールド・スタンバイ構成とも呼ばれ、PowerHA、TSAなどの High Availability 機能を提供するソフトウェアによって、アクティブ系のサーバーに障害が発生したときにはスタンバイ系のサーバーがIPアドレスやディスクを引き継ぎ、処理を継続します。この構成では、WASから見える情報であるDBサーバーのホスト名(IPアドレス)、ポート番号、データベース名など接続に必要な情報は変わりませんので、DBの引継ぎに際してDB接続情報の変更を考慮する必要はありません。

一方で、障害検知のタイミングからアクティブからスタンバイ機へのディスク、IP引継ぎに際して、ある程度時間がかかることが課題とされる構成であり、サービス再開までに通常数十秒程度はかかります。

このサービス停止時間を極力短くするために、また同時実行性を上げて高い処理性能を実現する目的から最右列のロード・バランス型クラスタリング構成も広く採用されています。

弊社製品としてはDB2 pureScale、他社製品で有名なものではOracle RACがあります。

アクティブ・アクティブ型、かつシェアード・ディスク型の構成をとり、1つのディスクに入っているデータを複数のデータベース・サーバーからアクセスできるため、特に読み取り系の処理では並列処理の効率がよい構成です。また、アクティブ・スタンバイ構成のデータベースに比べサーバー障害時の停止時間も極小化できる構成となります。WASから見ると接続先のデータベースサーバーが複数存在する構成となります。

アーキテクチャー	ホット・スタンバイ構成 (Active-Standby型)	ホット・スタンバイ構成 (Active-Active型※)
		 ★セカンダリーDBへの接続不可
DBのアーキテクチャー	アクティブ-ホット・スタンバイ型 製品例: DB2 HADR, Oracle Data Guard	アクティブ-アクティブ型※ (※:但し片系はReadOnly) 製品例: 同左
概要	プライマリー・サーバーからセカンダリー・サーバーへDBのデータ・ブロックやログを転送してDBサーバーを冗長化する仕組み	HADRのV9.7 FP1からのオプションとして、セカンダリーDBへの読取専用接続が可能になった
WASからのDB接続定義	災害切替であれば、サイト全体が切り替わるため接続定義の変更は通常不要。 サイト内切替の場合は、ホスト名(IP)切替対応が必要。(プライマリーとセカンダリーでIPアドレスが異なるため)	同左
DB障害時のサービス影響	テークオーバー中は全体停止となる 復旧時間目標は数十秒から1分程度	同左

© 2012 IBM Corporation

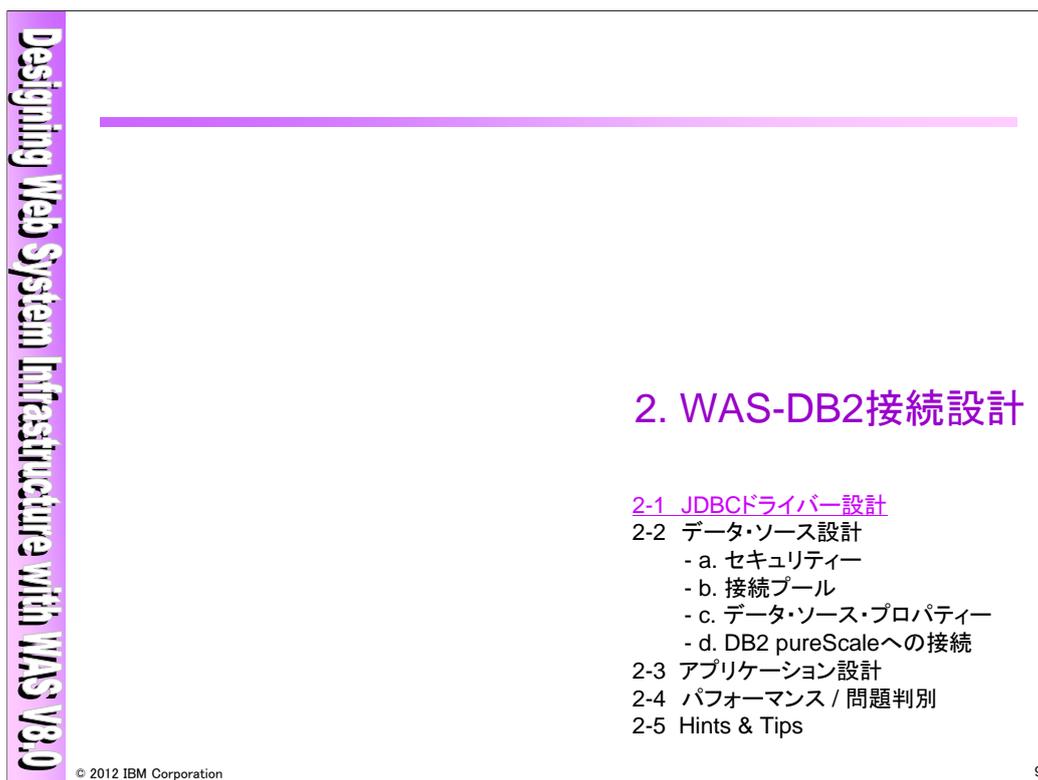
8

残る2つは、ホット・スタンバイ型構成で、IBMではDB2 HADR, OracleではOracle Data Guardと呼ばれる機能により実現されます。

主に災害対策を考慮した構成であり、データベースのデータ・ブロックを災害サイトのデータベースに直接送るか、データベース更新ログを災害サイトに送信し適用することで、本番サイトと同等のデータベースを災害サイトに保持する方式です。

基本的には災害対策用途で使われるテクノロジーですが、同一サイト内での障害対策として通常のHA構成(コールド・スタンバイ)の代わりに利用されるケースもあります。

この構成の場合、本番サイトと災害サイトとでListenするIPアドレスが異なるため、障害時の切り替えの際にはWAS側でデータ・ソース定義を変更するか、災害サイトのデータベース・サーバー側で、本番サイトと同じIPアドレスにつけかえるなどの対応が必要となります。



2章では、WASからIBM DB2 Database for Linux, UNIX, and Windows(以降、DB2)にアクセスするための設計・設定についてご説明致します。データベースとしてDB2を対象としていますが、他ベンダーのデータベースでもほとんど同様の設計・設定となります。(一部、WAS-DB2固有の機能がござります。) WAS-Oracle固有の設計・設定につきましては、3章をご確認下さい。

はじめに、JDBCドライバー設計についてご説明致します。JDBCドライバーは、管理コンソールから有効範囲、JDBCプロバイダー名、クラス・パス、ネイティブ・パス、実装クラス名などを設定し、新規作成して下さい。

詳しくは、巻末資料「DB接続設計-参考-」もあわせてご参照いただければと存じます。

Designing Web System Infrastructure with WAS V8.0

## WAS: JDBC プロバイダー

WebSphere software

表示: すべてのタスク

- ようこそ
- ☑ ガイド付きアクティビティ
- ☑ サーバー
- ☑ アプリケーション
- ☑ サービス
- ☑ リソース
  - スケジューラー
  - オブジェクト・プール・マネージャ
  - ☑ JMS
    - ☑ JDBC
      - JDBC プロバイダー
      - データ・ソース
      - データ・ソース (WebSphere Application Server V4)
    - ☑ リソース・アダプター
    - ☑ 非同期 Bean
    - ☑ キャッシュ・インスタンス
    - ☑ メール
    - ☑ URL
    - ☑ リソース環境

WAS構成情報(管理コンソールから設定する内容)

■JDBCプロバイダー

- ✓ JDBC Provider名
- ✓ 接続先DBMS種別 (DB2,Oracle, ..)
- ✓ JDBC DriverのFull Path
- ✓ JDBC Driver 実装のJavaクラス名
- ✓ クラスパス、ネイティブライブラリパス... etc

■データ・ソース

- ✓ データ・ソース名
- ✓ 紐付けるJDBC プロバイダー名
- ✓ JNDI名
- ✓ DataStoreHelperクラス名
- ✓ 接続先のホスト、ポート、DB名
- ✓ 認証別名 (UserID, password)
- ✓ 接続プール設定 (Timeout他)
- ✓ 接続テスト設定
- ✓ ステートメント・キャッシュ・サイズ
- ✓ カスタム・プロパティ、その他 ... etc

© 2012 IBM Corporation 10

WASからデータベース接続を行うにあたり、必要な設定項目を上の図に一覧しました。

設定項目は、大きく「JDBC プロバイダー」と「データ・ソース」に分かれます。

JDBC プロバイダーには、接続先DB種別、JDBCドライバや各種ライブラリー・ファイルのパス情報など、接続先のDBMSが同じであれば基本的に再利用可能な定義情報が含まれます。

データ・ソースには、接続先のDBサーバーホスト名、ポート番号、DB名や、データ・ソース自身がアプリケーションからどのような名前 (JNDI名) でlookupされるか、またどのユーザID/パスワードにて接続を行うか、等の詳細な接続情報が定義されます。

この章では、より大枠の定義情報であるJDBC プロバイダーの設定項目について説明します。

Designing Web System Infrastructure with WAS V8.0

## DB2のJDBCドライバー

- WASV8がサポートするDB2のJDBCドライバー
  - ◆ IBM Data Server Driver for JDBC and SQLJ (nonXA / XA) (Type 2/4)
    - JDBC3.0対応版 (db2jcc.jar)とJDBC4.0対応版 (db2jcc4.jar) に分かれる
  - ◆ DB2 Legacy CLI-based Type 2 JDBC DriversはWAS V7以降ではサポートされません
- JDBCプロバイダー選択指針
  - ◆ WAS V8 では JDBC 4.0 対応の db2jcc4.jar を推奨
- ドライバー・タイプ選択指針
  - ◆ WASとDB2が同一筐体/別筐体に関わらず、Type4 を選択
    - Type2は、OS上のDLLや専用ライブラリー経由でデータベースにアクセスするため、問題判別が煩雑となるケースがある
    - WASとDB2が同一筐体の場合、Type2の方がドライバー間の通信は高速であるが、トータルコストで比較するとほとんど差がない

プロバイダータイプ	JARファイル	ドライバーバージョン	JDBCサポート	Javaバージョン
JCC driver	db2jcc4.jar	JCC 4.0	JDBC4.0 and earlier	1.6
Universal driver	db2jcc.jar	JCC 3.5	JDBC3.0 and earlier	1.4

© 2012 IBM Corporation 11

WASV8がサポートするDB2のJDBCプロバイダーは、IBM Data Server Driver for JDBC and SQLJです。Legacy JDBCドライバーはWASV7以降ではサポートされませんのでご注意ください。

• Support for DB2 legacy CLI-based Type 2 JDBC Drivers is removed from IBM WebSphere Application Server Version 7.0

<http://www-01.ibm.com/support/docview.wss?uid=swg21316317>

• System Requirements for WebSphere Application Server V8.0 on IBM AIX

[http://www-01.ibm.com/support/docview.wss?uid=swg27021246#AIX\\_JDBC\\_Drivers\\_ww](http://www-01.ibm.com/support/docview.wss?uid=swg27021246#AIX_JDBC_Drivers_ww)

• WASV8.0 Information Center - 「JDBCプロバイダーの要約」

[http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.websphere.nd.multipatform.doc/info/ae/ae/udat\\_minreq.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.websphere.nd.multipatform.doc/info/ae/ae/udat_minreq.html)

• DB2V9.7 InformationCenter - 「Supported drivers for JDBC and SQLJ」

[http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=%2Fcom.ibm.db2.luw.apdv.java.doc%2Fsrc%2Ftpc%2Fimjcc\\_c0024189.html](http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=%2Fcom.ibm.db2.luw.apdv.java.doc%2Fsrc%2Ftpc%2Fimjcc_c0024189.html)

JDBC4.0にてisValid()メソッド等が様々な機能が追加されました。JDBC4.0の機能を使用する要件がある場合には、JCCドライバーをご選択下さい。

ドライバータイプは、今までは、WAS/DB2が同一筐体の場合には、パフォーマンスの観点からType2を推奨していました。しかしながら、スライド部分に記述した理由により、Type2に設定しなければいけない場合を除き、WAS/DB2が同一筐体/別筐体に限らずType4をご選択されても宜しいかと思えます。また、この考え方は、WASV8だけではなく、WASV7にも当てはまります。

db2jcc.jarとdb2jcc4.jarは、DB2製品コードのインストールディレクトリーsqllib/java (UNIX)、sqllib¥java (Windows)に保管されています。

Designing Web System Infrastructure with WAS V8.0

## (参考)一般的なJDBCドライバー

- 機能、特徴、パフォーマンス
  - ◆ Type 1 (JDBC-ODBC Bridge)
    - Protocol変換するのは、ODBCドライバーに依存する
    - 一般的には、最もパフォーマンスが悪い
  - ◆ Type 2 (Native API Partly)
    - アプリケーションからJNI経由で、直接APIを利用
    - 一般的には、直接APIを呼び出す為、機能が豊富で、比較的パフォーマンスが良い
  - ◆ Type 3 (Net-protocol pure Java)
    - JavaAppletで利用するためのJDBCドライバ
    - アプリケーションはWebサーバ上のリスナーを経由してDBサーバに接続
    - リスナーがデータベースにアクセスし、応答を返す
  - ◆ Type 4 (Native-protocol pure Java)
    - Javaのみでコーディング
    - ドライバーが、データベース固有の通信プロトコルに変換
    - 一般的には、データベース固有のプロトコルを利用する為、比較的パフォーマンスが良い
- 選択指針
  - ◆ 基本的に、データベース・ベンダーの指示に従って下さい

© 2012 IBM Corporation 12

一般的なJDBCドライバーのタイプ毎の機能、特徴、パフォーマンスをまとめると、上記スライド部分になります。JDBCドライバーはベンダー毎に提供されていますので、どれを選択するかについては、基本的に、各データベース・ベンダーの指示に従って下さい。DB2の場合にはパフォーマンスの観点から、アプリケーション・サーバーとデータベースが同一筐体の場合はType2を、別筐体の場合はType4を選択するのが良いと言われています。

Designing Web System Infrastructure with WAS V8.0

## JDBC Driverバージョンの指定方法

- WAS管理コンソール
  - ◆ リソース > JDBC プロバイダー > 新規作成
  - ◆ 「プロバイダー・タイプ」の選択により、使用されるJDBCドライバー・バージョンが決まる

■ プロバイダー・タイプ選択

新規 JDBC プロバイダーの作成

データベースのアクセスに必要な固有のベンダー JDBC ドライバ

有効範囲  
cells:darjeelingNode02Cell:nodes:darjeelingN...

\* データベース・タイプ  
DB2

\* プロバイダー・タイプ  
選択...

- DB2 Using IBM JCC Driver
- DB2 Universal JDBC Driver Provider
- DB2 UDB for iSeries (Native)
- DB2 UDB for iSeries (Toolbox)
- 非推奨ドライバーの表示...

★こちらがおすすめです！

■ JCC Driver 選択時に設定されるクラスパス (JDBC4.0)

```

${DB2_JCC_DRIVER_PATH}/db2jcc4.jar
${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cu.jar
${DB2_JCC_DRIVER_PATH}/db2jcc_license_cisuz.jar
${PUREQUERY_PATH}/pdq.jar
${PUREQUERY_PATH}/pdqgmt.jar

```

■ Universal JDBC Driver 選択時に設定されるクラス・パス (JDBC3.0)

```

${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar
${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cu.jar
${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cisuz.jar

```

(補足) db2jcc\_license\_\*.jar は DB2 V9.5 以降は不要となりました。

© 2012 IBM Corporation 13

このページでは、WASの管理コンソールでどのようにJDBCドライバーを指定するかをご説明します。

WAS管理コンソールから、JDBCドライバーを指定する場所は、「JDBCプロバイダー」作成時の「プロバイダー・タイプ」の選択画面となります。

ここで、「DB2 Using IBM JCC Driver」を選択してJDBCプロバイダーを作成すると、JDBC4.0対応のJDBCドライバー (db2jcc4.jar) にクラス・パスが設定されます。

「DB2 Universal JDBC Driver Provider」を選択すると、JDBC3.0対応のJDBCドライバーにクラス・パスが通ります。

なお、DB2 V9.5 よりJDBCドライバー利用時のライセンス・ファイルへのクラス・パス設定は、基本的に不要となりました。

ただし、DB2 for z/OS に接続する際や、クライアントのライセンスが必要なサーバーについては引き続き必要となります。

WebSphereの管理コンソールでは、デフォルトで画面キャプチャーの通りこれまで通りdb2jcc\_license\_xx.jarファイルのクラス・パスが通っています。基本的には、こちらは変更せずこのままご利用いただくので問題ありません。

Designing Web System Infrastructure with WAS V8.0

## DB2 V9.x にて提供されるJDBCドライバー

■ DB2 V9.1～V9.7 では下記のドライバーが提供されています。

WAS: JDBC Provider Type	JDBC Version	DB2 Version	DB2 8.2以前 *サポート終了	DB2 9.1	DB2 9.5	DB2 9.7
DB2 Legacy CLI-based Type 2 JDBC Drivers (サポート終了)	2.0	DB2 Legacy CLI-based Type 2 JDBC Drivers (db2java.zip)	サポート終了	非推奨 →		
DB2 Universal JDBC Driver Provider	2.0	DB2 Universal JDBC driver (db2jcc.jar)	サポート終了			
	3.0	DB2 Driver for JDBC and SQLJ (db2jcc.jar)		サポート有		
DB2 Using IBM JCC Driver	4.0	DB2 Data Server Driver for JDBC and SQLJ (db2jcc.jar)				サポート有 →
		DB2 Data Server Driver for JDBC and SQLJ (db2jcc4.jar)				サポート有 →

※ Universal / DB2 Driver / JCC / Data Server Driver は名前が異なるのみで、実体としては同じです。

© 2012 IBM Corporation 14

DB2 JDBCドライバーは、DB2バージョンアップに伴い、ドライバーの名称やドライバー・ファイルの名称が変わっています。

ピンクの枠で囲んでいるドライバーは、基本的に同じドライバー・モジュール(db2jcc.jar)をさしますが、DB2のバージョンにより名称が変わっています。

また、DB2 9.5 では、JDBC4.0対応済みのドライバーとして、db2jcc4.jar というドライバーが別途提供されるようになっています。

Designing Web System Infrastructure with WAS V8.0

### まとめ: JDBCドライバー設計

項目	選択肢	設計指針
プロバイダー・タイプ	<ul style="list-style-type: none"> <li>DB2 Using IBM JCC Driver (JDBC 4.0対応)</li> <li>DB2 Universal JDBC Driver Provider (JDBC3.0対応)</li> </ul>	JDBC4.0がサポートされる db2jcc4.jar (DB2 Using IBM JCC Driver) を推奨します。
実装タイプ	<ul style="list-style-type: none"> <li>接続プール・データ・ソース (nonXA)</li> <li>XAデータ・ソース (XA)</li> </ul>	2フェーズ・コミット処理を実施する場合は、XAを選択します。 2フェーズ・コミット処理を実施しない場合は、nonXAを選択します。
ドライバー・タイプ	<ul style="list-style-type: none"> <li>Type2</li> <li>Type4</li> </ul>	基本的に、データベース・ベンダーの指示に従って下さい。 DB2では、WASとDB2が同一筐体/別筐体に関わらず、Type4を推奨します。
DataStoreHelper (参考資料)	<ul style="list-style-type: none"> <li>DB2UniversalDataStoreHelper</li> <li>ユーザー定義</li> </ul>	JDBCドライバー毎に提供されているDataStoreHelperを選択して下さい。 エラー検出モデルは、「例外マッピング・モデルの使用」をデフォルト値として下さい。

JDBCドライバーの設定方法は、巻末資料「DB接続設計-参考-」をご参照下さい。

© 2012 IBM Corporation 15

JDBCドライバー設計について、その設計指針をまとめます。

Designing Web System Infrastructure with WAS V8.0

---

## 2. WAS-DB2接続設計

- 2-1 JDBCドライバー設計
- 2-2 データ・ソース設計
  - a. セキュリティー
  - b. 接続プール
  - c. データ・ソース・プロパティー
  - d. DB2 pureScaleへの接続
- 2-3 アプリケーション設計
- 2-4 パフォーマンス / 問題判別
- 2-5 Hints & Tips

© 2012 IBM Corporation 16

続きまして、データ・ソースについてご説明致します。

データ・ソースは、管理コンソールから、データ・ソース名や接続先データベースの情報(ポート番号、サーバー名、ドライバー・タイプ、データベース名)を設定し、作成します。

詳細は、巻末資料「DB接続設計-参考-」もあわせてご参照下さい。

Designing Web System Infrastructure with WAS V8.0

## WAS: データ・ソース

WebSphere, software

表示: すべてのタスク

- ようこそ
- ☑ ガイド付きアクティビティ
- ☑ サーバー
- ☑ アプリケーション
- ☑ サービス
- ☑ リソース
  - スケジューラー
  - オブジェクト・プール・マネージャ
  - ☑ JMS
    - ☑ JDBC
      - JDBC プロバイダー
      - データ・ソース
      - データ・ソース (WebSphere Application Server V4)
  - ☑ リソース・アダプター
  - ☑ 非同期 Bean
  - ☑ キャッシュ・インスタンス
  - ☑ メール
  - ☑ URL
  - ☑ リソース環境

WAS構成情報 (管理コンソールから設定する内容)

■JDBCプロバイダー

- ✓ JDBC プロバイダー名
- ✓ 接続先DBMS種別 (DB2,Oracle, ..)
- ✓ JDBC DriverのFull Path
- ✓ JDBC Driver 実装のJavaクラス名
- ✓ クラス・パス、ネイティブ・ライブラリー・パス.. etc

■データ・ソース

- ✓ データ・ソース名
- ✓ 紐付けるJDBC プロバイダー名
- ✓ JNDI名
- ✓ DataStoreHelperクラス名
- ✓ 接続先のホスト、ポート、DB名
- ✓ 認証別名 (UserID, password)
- ✓ 接続プール設定 (Timeout他)
- ✓ 接続テスト設定
- ✓ ステートメント・キャッシュ・サイズ
- ✓ カスタム・プロパティ、その他 ... etc

© 2012 IBM Corporation 17

データ・ソースは、管理コンソールから、データソース名や接続先データベースの情報(接続先のホスト名、ポート・データベース名、接続ユーザの情報その他)を設定し、作成します。

DB接続ユーザID/パスワードの指定方法や接続プールの設定について、ステートメント・キャッシュについて、また接続オブジェクトが有効であるかどうかの接続テストについての設定などをご紹介します。

Designing Web System Infrastructure with WAS V8.0

---

## 2. WAS-DB2接続設計

- 2-1 JDBCドライバー設計
- 2-2 データ・ソース設計
  - a. セキュリティー
  - b. 接続プール
  - c. データ・ソース・プロパティ
  - d. DB2 pureScaleへの接続
- 2-3 アプリケーション設計
- 2-4 パフォーマンス / 問題判別
- 2-5 Hints & Tips

© 2012 IBM Corporation 18

まずは、データ・ソース設計のセキュリティーについてご説明致します。

Designing Web System Infrastructure with WAS V8.0

## WAS-DB2間のセキュリティー設定

- WASにおけるDB接続ユーザID/パスワードの指定方法
  - ◆ WAS経由でデータベースに接続する際の認証情報 (ユーザー名、パスワード)
- DB2接続ユーザの変更 (DB2 Trusted Context)
- WAS ClientInformation API

Webアプリケーション・サーバー

DataSource

Application

Managed Connection

Connection Pool

JDBC Driver

DBサーバー

Data

Physical Connection

<<getConnection>>  
ユーザID/パスワードをどう渡す?

© 2012 IBM Corporation 19

ここからはセキュリティーとして、WASからデータベース・サーバーにユーザーID/パスワードを渡すための設定についてご紹介します。

Designing Web System Infrastructure with WAS V8.0

## JavaEEにおけるDB接続ユーザーID/パスワードの指定

- JavaEEコンテナは、DB、MQなどの各種リソースへの接続時に利用する認証情報【ユーザーID/パスワード】を指定する方法として、下記の2つ（1. 2.）を提供する必要があります。

<JavaEE Specより抜粋>

### EE.3.4.5 Resource Authentication Requirements

Resources within an enterprise are often deployed in security policy domains different from the security policy domain of the application component. The wide variance of authentication mechanisms used to authenticate the caller to resources leads to the requirement that a Java EE product provide the means to authenticate in the security policy domain of the resource.

**A Product Provider must support both of the following:**

**1. Configured Identity.**  
A Java EE container must be able to authenticate for access to a resource by using a principal and authentication data specified by a Deployer at deployment time. The authentication must not depend in any way on data provided by the application components. Providing for the confidential storage of the authentication information is the responsibility of the Product Provider.

**2. Programmatic Authentication.**  
The Java EE product must provide for specification of the principal and authentication data for a resource by the application component using appropriate APIs. The application may obtain the principal and authentication data through a variety of mechanisms, including receiving them as parameters, obtaining them from the component's environment, and so forth.

JavaEEコンテナにあらかじめ定義された認証情報（ユーザーID/パスワードのペア）を利用する方法

アプリケーション・コンポーネントからJavaのAPI経由で認証情報を渡す方法

20

WASの説明に入る前に、JavaEEの仕様を軽くおさえておきたいと思います。（プレゼンチャートの小さい字の部分を読んでいただく必要はありません。）

JavaEEの仕様では、データベースなどリソースに対して、JavaEEコンテナから接続を行う上で、2通りの認証の仕組みをサポートする必要があると定められています。

1つ目は「1. Configured Identity」、JavaEEコンテナにあらかじめ定義された認証情報を利用する方法です。

ここでいう認証情報とは、接続ユーザのIDとパスワードのセットのことです。

2つ目は「2. Programmatic Authentication」、アプリケーション・コンポーネントからJavaのAPIを経由して認証情報を渡す方法です。

JavaのAPI経由で渡すとは、JDBCならばgetConnectionメソッドの引数で渡す方法が典型的です。

Designing Web System Infrastructure with WAS V8.0

## WAS V8におけるリソース認証情報の指定

- WASでは、コンテナ管理認証、アプリケーション管理認証を利用します。
- アプリケーション管理認証では、ユーザID/パスワードを渡す方法が複数提供されます。

<JavaEE Specより抜粋>

### EE.3.4.5 Resource Authentication Requirements

Resources within an enterprise are often deployed in security policy domains different from the security policy domain of the application component. The wide variance of authentication mechanisms used in these domains leads to the requirement that a Java EE product provider must support both of the following authentication mechanisms:

A Product Provider must support both of the following authentication mechanisms:

- 1. Configured Identity.**  
A Java EE container must be able to authenticate for an application component using a principal and authentication data specified by a Deployment Descriptor at deployment time. The authentication must not depend in any way on information provided by the application components. Providing for the configuration of the authentication information is the responsibility of the Product Provider.
- 2. Programmatic Authentication.**  
The Java EE product must support authentication data being passed to the application component using appropriate API methods. The authentication data is then used as parameters to the authentication mechanism and so forth.

**WAS【コンテナ管理認証】**

データソース定義の「コンテナ管理認証別名」が利用される

web.xml <res-auth>にContainerを指定することでコンテナ管理認証を行うアプリケーションとして扱われる

**WAS【アプリケーション管理認証】**（下記1.2.3.の順番に取得される）

1. getConnectionメソッドに渡されるユーザー ID、パスワード  
【注意】 接続先のデータベースと認証情報が一致しない場合には、接続プールが再利用されない
2. データソース内の「コンポーネント管理認証別名」
3. データソースのカスタム・プロパティのユーザー名、パスワード

web.xml <res-auth>にApplicationを指定することでアプリケーション管理認証を行うアプリケーションとして扱われる

© 2012 IBM Corporation

WASでは、「1. Configured Identity」を「コンテナ管理認証」として、また「2. Programmatic Authentication」を「アプリケーション管理認証」として提供しています。

お客様のアプリケーションがコンテナ管理認証とコンポーネント管理認証のどちらを使用するかについては、アプリケーションのデプロイメント記述子(web.xml)の<res-auth>に設定します。

<res-auth>にContainerを設定した場合、WASデータソースに定義された「コンテナ管理認証別名」という定義情報が、ユーザーID/パスワードとして利用されます。この「認証別名」というWASの定義情報については、次のページにて説明いたします。

<res-auth>にApplicationを設定した場合、WASはスライド部分1～3の順番で認証方法が評価されます。1の方法は、アプリケーション内のgetConnectionの引数にて認証情報を指定しますが、接続先DBと認証情報が一致しない場合には、接続プールに蓄積されている接続オブジェクトを再利用することが出来ません。データベースとの接続確立や接続終了処理は負荷がかかる処理ですので、認証情報が複数ある場合等には、1の方式を利用することはパフォーマンスの観点から難しいかと思えます。従いまして、基本的には2の方式にてご設定下さい。

Designing Web System Infrastructure with WAS V8.0

## J2C認証別名(J2C認証データ)

- WASでは、DB接続ユーザー、パスワードを”J2C認証別名(エイリアス)”として管理する
  - ◆ 別名
  - ◆ DB接続ユーザー
  - ◆ DB接続ユーザーのパスワード
- WAS管理コンソール
  - ◆ リソース>JDBC>データ・ソース>[データ・ソース名]>JAAS - J2C 認証データ>新規作成

■ WAS J2C認証別名作成画面

一般プロパティ

\* 別名  
testAuth

\* ユーザー ID  
user01

\* パスワード  
password

説明  
User for test

適用 OK リセット キャンセル

■ WASアプリケーション用ユーザー:

- ユーザID=user01
- パスワード=password

DBサーバー

© 2012 IBM Corporation 22

WASでは、データベース接続ユーザID、パスワードのセットを「J2C認証別名(エイリアス)」と呼ばれる定義情報として管理します。

認証別名には、ユーザーID、パスワードと、そのユーザID/パスワードのセットをなんという名前かで管理するかをあらわす別名(エイリアス)をあわせて定義します。

J2C認証別名は、WASの管理コンソールから作成します。

別名には、DB接続ユーザのIDとパスワードを設定し、保管します。

Designing Web System Infrastructure with WAS V8.0

## データ・ソースとJ2C認証別名のマッピング (1/2)

- WASからのDB接続ユーザーを指定する方法①:
  - ◆ **WASデータ・ソースに対し、J2C認証別名を対応づける**
    - ▶ コンテナ管理認証とする場合には、データ・ソースのコンテナ管理認証別名を設定しておく
    - ▶ アプリケーション管理認証を想定する場合、データ・ソースのコンポーネント管理認証別名を設定しておく
    - ▶ 認証別名をWASのデータ・ソースへ設定することによって、本来そのデータ・ソースを使用すべきではないアプリケーションからもデータ・ソースが使用されてしまう可能性があり、セキュリティの観点から好ましくない場合がある(非推奨)

23

前頁の要領で作成した認証別名は、作成しただけではどのデータ・ソース定義にも利用されません。WASからのデータベース接続においてこの認証別名(=ユーザーID、パスワード情報)を利用するには、どの認証別名の情報で接続を行うかを指定する必要があります。

WASでは、WAS上のデータソース定義として直接どの認証別名を使うか定義することもできますし、次のページでご紹介する、アプリケーションにて指定する方法も選択することができます。

このページの図で示しているのは、WASデータ・ソースの定義情報として、あらかじめ作成しておいた認証別名を紐付ける方法です。

WASのデータ・ソースのプロパティとして、DB接続時にデフォルトで利用する認証別名を指定することができます。

Designing Web System Infrastructure with WAS V8.0

★こちらがおすすめです！

## データ・ソースとJ2C認証別名のマッピング (2/2)

- WASからのDB接続ユーザーを指定する方法②:
  - ◆ 「コンテナ管理認証別名」としてアプリケーションのリソース参照に設定
    - コンテナ管理認証は、リソース参照のコンテナ管理認証別名を使用
    - 認証別名をアプリケーションのリソース参照にマッピングすれば、そのリソース参照を使用するモジュール内でしか使用されない

■ WAS アプリケーション 認証別名の指定 (アプリケーションデプロイ Wizardより)

認証方式の指定:

なし。

デフォルト・メソッドの使用 (多対 1 のマッピング)

認証データ入力  
darjeelingNode00/testAuth

トラストド接続の使用 (1対 1 のマッピング)

認証データ入力  
選択...

カスタム・ロビン構成を使用  
アプリケーション・ロビン構成

■ WAS J2C認証別名

一般プロパティ

\* 別名  
testAuth

\* ユーザー ID  
user01

\* パスワード  
\*\*\*\*\*

説明  
user for test

適用 OK リセット キャンセル

WAS Application Data Source (Connect on Pool, JDBC Driver)

DBサーバー

■WASアプリケーション用ユーザー:  
●ユーザーID=user01  
●パスワード=password

© 2012 IBM Corporation 24

前のページで示した、WASのコンテナ管理認証別名を利用する方法は、前ページのスライド部分に記述しています通り(※)、お客様のセキュリティ要件と合致しない場合があります。

その場合の対応として、こちらのページでご紹介するように、アプリケーションのリソース参照にコンテナ管理認証 + コンテナ管理認証別名として認証情報を設定する方法をおすすめいたします。

(※)認証別名をWASのデータ・ソースへ設定することによって、本来そのデータ・ソースを使用すべきではないアプリケーションからもデータ・ソースが使用されてしまう可能性があり、セキュリティの観点から好ましくない場合がある

Designing Web System Infrastructure with WAS V8.0

## ClientInformationAPI

- Webアプリケーションにクライアント情報を設定し、その情報をデータベースに受け渡すことができる機能 (WASV6以降)
- 設定方法
 

```
import com.ibm.websphere.rsadapter.WSConnection;
(省略)
con = (WSConnection) ds.getConnection();
Properties props = new Properties();
props.setProperty(WSConnection.CLIENT_ID, userid);
props.setProperty(WSConnection.CLIENT_LOCATION, location);
props.setProperty(WSConnection.CLIENT_ACCOUNTING_INFO, accounting);
props.setProperty(WSConnection.CLIENT_APPLICATION_NAME, appname);
props.setProperty(WSConnection.CLIENT_OTHER_INFO, other_info);
props.setProperty(WSConnection.OTHER_CLIENT_TYPE, client_type);
con.setClientInformation(props);
```

WSConnectionをimportする

ここにクライアント情報を設定する
- 適用例
  - ◆ 監査ログ
    - WASとDB2の監査ログを一致することができる
  - ◆ ワークロード管理
    - Webシステム全体でのワークロード管理を実現できる

© 2012 IBM Corporation 25

ClientInformationAPIとは、Webアプリケーションにクライアント情報を設定し、その情報をデータベースに受け渡すことができる機能です。WASV6以降でサポートされます。

Webアプリケーション・コード内に、WSConnectionをimportし、setPropertyメソッドにてクライアント情報を明示的に設定して下さい。

この機能の適用例としては、監査ログやワークロード管理が考えられ、詳細は次ページ以降で説明致します。

•WASV8.0 Information Center - 「setClientInformation(Properties) API によるクライアント情報の設定」

[http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tdat\\_clientinfotask.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tdat_clientinfotask.html)

•DB2V9.7 InformationCenter - 「WLM\_SET\_CLIENT\_INFO ストアードプロシージャ」

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.sql.rtn.doc/doc/r0053116.html>

Designing Web System Infrastructure with WAS V8.0

## ClientInformationAPI適用例【監査ログ】

### Webアプリケーション設定例

```

import com.ibm.websphere.rsadapter.WSConnection;
con = (WSConnection) ds.getConnection();
Properties props = new Properties();
props.setProperty(WSConnection.CLIENT_ID, Jack);
con.setClientInformation(props);
                    
```

### DB2監査ログ出力例

```

client userid=Jack;
statement text=SELECT * FROM
db2inst1.tableA WHERE col1 = ?;
statement isolation level=RS;
                    
```

ClientInformationAPIを使用していないWAS/DB2システムの場合、DB2へアクセスするユーザーはデータ・ソースに定義された代表ユーザー(db2inst1)となります。このため、WAS側では、Maryさん、Jackさんというようにクライアントリクエスト毎に監査ログが取得できても、DB2側では全てdb2inst1さんの監査ログになってしまい、厳密な意味でWASとDB2の監査ログを一致できないという課題がありました。それに対してClientInformationAPIを使用すると、ClientInformationAPIで指定した情報を、DB2側の監査ログに出力されることができるようになります。



Designing Web System Infrastructure with WAS V8.0

### まとめ: データ・ソース設計 ～セキュリティ

項目	選択肢	設計指針
認証方法	<ul style="list-style-type: none"> <li>・コンテナ管理認証 + コンテナ管理認証別名</li> <li>・アプリケーション管理認証 + コンポーネント管理認証別名</li> </ul>	「アプリケーション管理認証 + コンポーネント管理認証別名」を推奨します。
ユーザーID、パスワードの指定箇所	<ul style="list-style-type: none"> <li>・getConnectionメソッドに渡されるユーザー ID、パスワード</li> <li>・接続ファクトリー、またはデータ・ソース内のコンポーネント管理認証別名</li> <li>・データ・ソースのカスタム・プロパティのユーザー名、パスワード</li> </ul>	認証別名にユーザーID、パスワードを設定し、コンポーネント管理認証別名に指定することを推奨します。
ClientInformationAPIによる監査ログ取得	<ul style="list-style-type: none"> <li>・ClientInformationAPI</li> <li>・DB2 監査ログ</li> </ul>	WAS側とDB2側の監査ログ内容を(厳密に)一致させたい場合に、設定することを検討して下さい。
DB2 Trusted Context (参考)	-	<p>以下のような、DB2のオブジェクト管理を実施する要件がある場合に設定することをご検討下さい。</p> <ul style="list-style-type: none"> <li>・ユーザー毎の監査</li> <li>・特権や権限の適切な付与</li> </ul>

セキュリティの設定方法は、巻末資料「DB接続設計-参考-」をご参照下さい。

© 2012 IBM Corporation 28

データ・ソース設計のセキュリティについて、その設計指針をまとめます。

Designing Web System Infrastructure with WAS V8.0

---

## 2. WAS-DB2接続設計

- 2-1 JDBCドライバー設計
- 2-2 データ・ソース設計
  - a. セキュリティー
  - b. 接続プール
  - c. データ・ソース・プロパティ
  - d. DB2 pureScaleへの接続
- 2-3 アプリケーション設計
- 2-4 パフォーマンス / 問題判別
- 2-5 Hints & Tips

© 2012 IBM Corporation 29

続きまして、データ・ソースの接続プールの設計についてご説明致します。

Designing Web System Infrastructure with WAS V8.0

## WAS: 接続プール設定



WebSphere, software

表示: すべてのタスク

- ようこそ
- ☑ ガイド付きアクティビティ
- ☑ サーバー
- ☑ アプリケーション
- ☑ サービス
- ☑ リソース
  - スケジューラー
  - オブジェクト・プール・マネージャ
  - ☑ JMS
    - ☑ JDBC
      - JDBC プロバイダー
      - データソース
      - データソース (WebSphere Application Server V4)
  - ☑ リソース・アダプター
  - ☑ 非同期 Bean
  - ☑ キャッシュ・インスタンス
  - ☑ メール
  - ☑ URL
  - ☑ リソース環境

WAS構成情報(管理コンソールから設定する内容)

■JDBCプロバイダー

- ✓ JDBC プロバイダー名
- ✓ 接続先DBMS種別 (DB2,Oracle, ...)
- ✓ JDBC DriverのFull Path
- ✓ JDBC Driver 実装のJavaクラス名
- ✓ クラス・パス、ネイティブ・ライブラリー・パス.. etc

■データ・ソース

- ✓ データ・ソース名
- ✓ 紐付けるJDBC プロバイダー名
- ✓ JNDI名
- ✓ DataStoreHelperクラス名
- ✓ 接続先のホスト、ポート、DB名
- ✓ 認証別名 (UserID, password)
- ✓ 接続プール設定 (Timeout他)
- ✓ 接続テスト設定
- ✓ ステートメント・キャッシュ・サイズ
- ✓ カスタム・プロパティ、その他 ... etc

© 2012 IBM Corporation 30

ここでは、

- WAS,DB2リソースの再利用の考え方について
  - WAS接続プールのライフサイクル
  - WAS接続プールの設計指針
  - WAS接続プールの管理・監視
- についてご説明いたします。

Designing Web System Infrastructure with WAS V8.0

## WAS / DB2リソースの有効活用

- プール機能の比較
  - ◆ WASの接続プール
    - ▶ アプリケーションによる接続 / 切断の負荷を軽減する
    - ▶ データベースに対する同時並列処理数を制限する
    - ▶ Prepared Statement Cacheの機能を持つ
  - ◆ DB2のエージェント・プール
    - ▶ 接続 / 切断を繰り返すアプリケーションによる、接続開始時のdb2agentの起動負荷を減らす
    - ▶ db2agentをプールに戻す時に、メモリーを一部残して解放する
      - メモリーに残す値は、DB2MEMMAXFREE (レジストリ変数) に設定する (デフォルト:8MB)
      - プールに保持すべきエージェントの数は、NUM\_POOLAGENTSで設定する (デフォルト:100 Automatic)

The diagram illustrates the connection flow. On the left, three 'Application' boxes are shown. Arrows point from these applications to the 'WebSphere Application Server' box. Inside this server, there is a 'DataSource' box containing a 'Connection Pool' and a 'JDBC Driver'. Arrows from the 'Connection Pool' point to the 'DB2 Server' box. The 'DB2 Server' contains an 'エージェント・プール' (Agent Pool) with several 'agent' boxes, some of which are labeled 'idle agent'. A label '接続' (Connection) is placed between the WAS and DB2 components.

### ・WASの接続プール

WAS起動時にはデータベースへの接続は発生しません。ユーザーからのDBアクセスの処理要求により接続を行います。複数ユーザーが同時にDB接続処理を要求すると、その処理数分、接続を行い、処理終了時に接続を開放せずにそのまま保持します。また、未使用タイムアウトなどによって、接続プール内の接続を制御することができます。

### ・DB2のエージェント・プール

DB2では、エージェント・プロセスがクライアントからの要求を処理します。

エージェントの作成負荷を軽減するために、使用済みのエージェントをIDLE状態で保持する仕組みとしてエージェント・プールを利用しています。

クライアントからの接続が切断されると、DB2ではエージェント・プロセスをプールに保持し、それまで使用していたプライベート・メモリーの一部を解放します。(AIXではDB2MEMDISCLAIMおよびDB2MEMMAXFREEレジストリ変数によって設定します。)そして、再び接続要求があった時に、エージェント・プールからエージェント・プロセスを取り出し、再利用します。

WASとDB2を使用するシステムでは、WASの接続プール数、DB2のエージェント数、エージェント・プール数を考慮する必要があります。

### 【補足】

DB2 V9.5以降とそれ以前では、DB2 DBM構成パラメータNUM\_POOLAGENTSの意味が異なります。

NUM\_POOLAGENTS DBM構成パラメーターの変更について(DM-05-021)

<https://www-304.ibm.com/support/docview.wss?uid=jpn1J1008183>

Designing Web System Infrastructure with WAS V8.0

## 接続オブジェクトのライフサイクル

- Managed Connectionのステータス
  - ◆ DoesNotExist
    - 物理接続が無い
  - ◆ InUse
    - 物理接続が存在し、利用されている
  - ◆ InFreePool
    - 物理接続が存在するが、利用されていない
- ステータス遷移のタイミング
  - (1) DoesNotExist → InUse
    - 接続プール内にInFreePoolの接続オブジェクトがない時に、`getConnection()`
  - (2) InUse → InFreePool
    - `connection.close()`
  - (3) InFreePool → InUse
    - 接続プール内にInFreePoolの接続オブジェクトがある時に、`getConnection()`
  - (4) InUse → DoesNotExist
    - `StaleConnectionException`が発生
  - (5) InFreePool → DoesNotExist
    - 未使用タイムアウト、経過タイムアウトが発生
    - Purge PolicyがEntirePoolに設定されている場合、`StaleConnectionException`が発生

The diagram, titled 'Connection Pool', shows three states: DoesNotExist (top left), InFreePool (top right), and InUse (bottom center). Transitions are numbered: (1) DoesNotExist to InUse; (2) InUse to InFreePool; (3) InFreePool to InUse; (4) InUse to DoesNotExist; (5) InFreePool to DoesNotExist.

© 2012 IBM Corporation
32

接続オブジェクトは、以下の3種類のステータスがあります。

- DoesNotExist : 物理接続がない状態で無効のコネクション
- InUse : 物理接続を利用している状態
- InFreePool : プールにもどされた接続で未使用の状態

また、`StaleConnectionException`が発生すると、InUse、およびInFreePoolの接続オブジェクトはDoesNotExistに遷移します。

Designing Web System Infrastructure with WAS V8.0

## 接続プールの設定

- 接続タイムアウト
  - ◆ 接続待機状態から、ConnectionWaitTimeoutExceptionが発生するまでの間隔
- 最大接続数
  - ◆ プール可能な接続の最大数
- 最小接続数
  - ◆ プールに保持される接続の最小数
- リープ時間
  - ◆ プール保守スレッドの実行される間隔
- 未使用タイムアウト
  - ◆ 未使用接続に対して、接続を破棄する間隔(最小接続数まで破棄)
- 経過タイムアウト
  - ◆ 未使用接続に対して、全ての接続を破棄する間隔
- パージ・ポリシー
  - ◆ StaleConnectionExceptionが発生した場合の接続をパージする方法

33

接続タイムアウトは、アプリケーションが、dataSource.getConnection()を発行してから、接続オブジェクトを取得するまでの、最大待ち時間を設定します。接続タイムアウトが発生すると、アプリケーションにはConnectionWaitTimeoutExceptionが返ります。

最大接続数は、接続プールに作成することのできる、ManagedConnectionの最大数を設定します。最大接続数に達していて全ての接続がInUseの時に、接続タイムアウトが経過してもInFreePoolに戻る接続がない場合は、ConnectionWaitTimeoutExceptionが発生します。

最小接続数は、未使用タイムアウトで接続プール内の接続オブジェクトを解放するとき、最低限接続プールに残したい接続数を設定します。

リーブ時間は、接続オブジェクトを監視し、タイムアウトをチェックする間隔を設定します。リーブ処理のタイミングにて接続オブジェクトが解放されます。

未使用タイムアウトは、InFreePoolの接続オブジェクトを解放する間隔を設定します。また、接続オブジェクトの数が、最小接続数になるまで解放します。

経過タイムアウトは、接続オブジェクトを解放する間隔を設定します。この設定は、接続オブジェクトの数が0になるまで解放し、接続オブジェクトがInUseの場合には次にInFreePoolになるタイミングで解放します。

パージ・ポリシーは、StaleConnectionExceptionが発生した時に、InFreePoolの接続オブジェクトの取り扱い方を設定します。EntirePool(デフォルト値)を設定すると、InFreePoolの接続オブジェクトを全て解放します。FailingConnectionOnlyを設定すると、StaleConnectionExceptionが発生した接続オブジェクトのみを解放します。

Designing Web System Infrastructure with WAS V8.0

## 接続プールの設計指針

- 最大・最小接続数、未使用タイムアウトにより、定期的にWASからのDB接続を切断し、メモリーを解放する
  - ◆ WASの接続プールにより接続が切れないと、db2agentはメモリーを解放できない
- 最大接続数
  - ◆ ピーク時の1アプリケーション・サーバーのDBアクセス数を設定する
    - 接続先データベースの許容量を超えないように設定する
    - 縮退率とデータベース・サーバーの負荷を考慮する
- 最小接続数
  - ◆ 定常状態の接続数を設定する
  - ◆ 定期的にリソースを解放したい場合は、経過タイムアウトを設定する

© 2012 IBM Corporation 34

接続プールの設計指針を、まとめます。

上記を参考にして初期値を設定し、Tivoli Performance Viewer(以降、TPV)等により接続プールの使用状況を確認し、チューニングを実施して下さい。

Designing Web System Infrastructure with WAS V8.0

## 接続プールの管理

- 状況の確認
  - ◆ Tivoli Performance Viewer
  - ◆ 管理コンソール
  - ◆ wsadminコマンド
- パージ ~ 接続オブジェクトの破棄
  - ◆ 管理コンソール
  - ◆ wsadminコマンド
- 制御 ~ 接続プールの休止と再開
  - ◆ 管理コンソール
  - ◆ wsadminコマンド



```

$AdminControl invoke $SubjectName showPoolContents
            
```

```

$AdminControl invoke $Subjectname purgePoolContents normal
$AdminControl invoke $Subjectname purgePoolContents immediate
            
```

```

$AdminControl invoke $Subjectname pause
$AdminControl invoke $Subjectname resume
            
```

© 2012 IBM Corporation
35

接続プールの状況を確認する方法として、TPV、管理コンソール、wsadminがあります。管理コンソールでは、接続先のデータベースの状態をACTIVE / PAUSE / NOT\_ACCESSEDで判断することができます。wsadminコマンドでは、接続プールの設定値、接続数、接続プールに蓄積されている接続数等を確認することができます。出力例は、下記リンク先をご確認下さい。

・WASV8.0 Information Center - 「例: wsadmin を使用して MBean 接続ファクトリーおよびデータソースにアクセスする」

[http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/rdat\\_wsadminex.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/rdat_wsadminex.html)

管理コンソール、およびwsadminコマンドにて、任意に接続オブジェクトの破棄、休止と再開を実行することができます。任意に接続オブジェクトの破棄を行った場合、InFreePoolの接続オブジェクトは、即座に破棄されましたが、InUseの接続オブジェクトは、トランザクションの終了とコネクションのクローズを待ってから、該当の接続オブジェクトを破棄します。

・接続オブジェクトを破棄した時のWASTレース出力

```

Connection marked to be destroyed. Waiting for transaction end and connection close -
MCWrapper id 20122012 Managed connection WSRdbManagedConnectionImpl@20462046
State:STATE_TRAN_WRAPPER_INUSE Start time inuse Fri Dec 19 19:36:10 JST 2008 Time inuse
27 (seconds)
            
```

また、接続プールをPAUSEしたタイミングでDBアクセスを行うと、以下のSQLExceptionが発生し、DBアクセスすることが出来ません。

```

[1/22/09 10:54:00:750 GMT+09:00] 00000024 SystemErr R java.sql.SQLException: The pool is
currently paused. Resume the pool to start processing connections.
            
```

Designing Web System Infrastructure with WAS V8.0

### まとめ: データソース設計 ~ 接続プール

項目	選択肢	設計指針
接続プール	<ul style="list-style-type: none"> <li>・接続タイムアウト</li> <li>・最大接続数</li> <li>・最小接続数</li> <li>・リーブ時間</li> <li>・未使用タイムアウト</li> <li>・経過タイムアウト</li> <li>・パージ・ポリシー</li> </ul>	<ul style="list-style-type: none"> <li>・最大 / 最小接続数、未使用タイムアウトにより、定期的にWASからのDB接続を切断し、メモリーを解放して下さい。</li> <li>・最大接続数</li> </ul> <p>ピーク時の1アプリケーション・サーバーのDBアクセス数を設定して下さい。</p> <ul style="list-style-type: none"> <li>・最小接続数</li> </ul> <p>定常状態の接続数を設定して下さい。</p> <p>定期的にリソースを解放したい場合は、経過タイムアウトを設定して下さい。</p> <ul style="list-style-type: none"> <li>・未使用タイムアウト</li> </ul> <p>初期値として、デフォルト値を使用して下さい。</p>
サージ保護設計 (参考資料)	接続処理による負荷の軽減	DBサーバーに同時に大量の接続要求が行われ、接続処理効率が悪化している場合、設定することを検討して下さい。
滞留タイマー設計 (参考資料)	DBサーバー過負荷状態におけるリクエストの抑制	DBサーバーが過負荷に陥っており、更に処理を要求すると、全てのリクエストの処理効率が低下する想定される場合、設定することを検討して下さい。

サージ保護・滞留タイマーについては、巻末資料「DB接続設計-参考-」をご参照下さい。

© 2012 IBM Corporation 36

データソース設計の接続プールについて、その設計指針をまとめます。

Designing Web System Infrastructure with WAS V8.0

---

## 2. WAS-DB2接続設計

- 2-1 JDBCドライバー設計
- 2-2 データソース設計
  - a. セキュリティー
  - b. 接続プール
  - c. データソースプロパティ
  - d. DB2 pureScaleへの接続
- 2-3 アプリケーション設計
- 2-4 パフォーマンス / 問題判別
- 2-5 Hints & Tips

© 2012 IBM Corporation 37

続きまして、各種のデータソース・プロパティについてご説明いたします。

Designing Web System Infrastructure with WAS V8.0

## WAS: データ・ソース・プロパティ



WebSphere, software

表示: すべてのタスク

- ようこそ
- ☑ ガイド付きアクティビティ
- ☑ サーバー
- ☑ アプリケーション
- ☑ サービス
- ☑ リソース
  - スケジューラー
  - オブジェクト・プール・マネージャ
  - ☑ JMS
    - ☑ JDBC
      - JDBC プロバイダー
      - データ・ソース
      - データ・ソース (WebSphere Application Server V4)
    - ☑ リソース・アダプター
    - ☑ 非同期 Bean
    - ☑ キャッシュ・インスタンス
    - ☑ メール
    - ☑ URL
    - ☑ リソース環境

WAS構成情報(管理コンソールから設定する内容)

■JDBCプロバイダー

- ✓ JDBC プロバイダー名
- ✓ 接続先DBMS種別 (DB2,Oracle, ..)
- ✓ JDBC DriverのFull Path
- ✓ JDBC Driver 実装のJavaクラス名
- ✓ クラス・パス、ネイティブ・ライブラリー・パス.. etc

■データ・ソース

- ✓ データ・ソース名
- ✓ 紐付けるJDBC プロバイダー名
- ✓ JNDI名
- ✓ DataStoreHelperクラス名
- ✓ 接続先のホスト、ポート、DB名
- ✓ 認証別名 (UserID, password)
- ✓ 接続プール設定 (Timeout他)
- ✓ 接続テスト設定
- ✓ ステートメント・キャッシュ・サイズ
- ✓ カスタム・プロパティ、その他 ... etc

ここからはデータ・ソース・プロパティのご紹介です。

データソース・プロパティには、プリペア・ステートメント・キャッシュやDB2拡張データ・ソース機能など他にもいくつかの設定項目がありますが、ここではDBサーバーやネットワークの障害が発生して、WAS上の接続オブジェクトが失効した際のハンドリングの設定について焦点をあてて、設定や挙動を詳しくご説明していきたいと思ます。

プリペア・ステートメント・キャッシュ等、その他のデータソース・プロパティについて、巻末資料「DB接続設計 -参考-」にまとめておりますので、あわせてご一読ください。

WAS V8.0 によるWebシステム基盤設計ワークショップ

38

Designing Web System Infrastructure with WAS V8.0

## 接続オブジェクト失効時の対策

- WAS上の接続オブジェクトが失効する主な原因
  - ◆ WAS-DB間のネットワーク障害、DB障害、DBサーバーの再起動
- アプリケーションへの影響
  - ◆ StaleConnectionExceptionが発生する

- 対応策
  - ～ DB復旧後の最初のアクセスでExceptionを発生させない～
  - ◆ 対策1 : getConnection時に自動的に接続検証を行う設定
  - ◆ 対策2 : アプリケーション・コーディングによる接続障害時への対応

★ポイント: 対策1, 2 を併用して、DB接続エラーの発生確率を下げる !!

★考慮点 : 利用するJDBCドライバーのバージョンによって、実装方法が異なります。

© 2012 IBM Corporation 39

WAS-DB間のネットワーク障害、DB自体の障害、DBサーバーの再起動等が発生すると、WASの接続プールに蓄積されている接続オブジェクトが失効します。この失効した接続オブジェクトは、DBが復旧しても、WASを再起動するかページされない限り、接続プール内に残り続けます。この状況において、アプリケーションが接続プール内にある接続オブジェクトを取得しようとした場合、StaleConnectionExceptionが発生します。

DB復旧後の最初のアクセスでExceptionを発生させない対応として、以下が考えられます。

- ・管理コンソール:接続検証プロパティ
  - JDBC 4.0の場合 JDBCドライバー(isValid)による検証
  - JDBC 3.0の場合 SQL照会による検証 (V7では非推奨)
- ・アプリケーション実装
  - JDBC 4.0の場合 JDBCドライバー(isValid)による検証
  - JDBC 3.0の場合 StaleConnectionException発生をcatchし、リトライする

WASV6.1までは、StaleConnectionException発生をcatchし、リトライする方法が推奨されていました。WASV7/V8ではJDBC4.0がサポートされ、JDBC4.0の機能であるisValidメソッドを使用することが可能です。

Designing Web System Infrastructure with WAS V8.0

### 対策1: 接続取得時の接続検証【JDBC 4.0 vs 3.0】

- JDBC 4.0 環境での自動接続検証(接続検証プロパティー)**
  - ◆ 接続の取得時にisValid メソッドによる検証を行い、有効な接続であることを確認

★こちらがおすすめ！

- JDBC 3.0 環境での自動接続検証(SQL検証)**
  - ◆ 接続の取得時、ダミーのSQLを実行して接続オブジェクトが有効なものであることを確認した上で、実際のSQLを実行
  - ◆ DBサーバーに2回Queryが実行されるため、JDBC 4.0の方式に比べ性能上不利

© 2012 IBM Corporation 40

getConnection時に接続が有効であるかを確認する場合の動作を、JDBC 4.0 (isValid)とJDBC3.0 (SQL検証)とで比較してみましょう。

JDBC4.0対応のJDBCドライバーが利用できる環境にて、管理コンソールで「接続検証プロパティー」を設定しておく、getConnection()実行時に自動的に接続の検証が行われます。そして接続オブジェクトが有効であると判定されると、実際のSQLが実行されます。一方で、JDBC3.0/SQL検証では、ダミーのSQLで接続オブジェクトの有効性を確認後、実際のSQLを発行する仕組みとなるため、ダミーのSQLを実行するオーバーヘッドが発生するという課題があります。なお、検証時に実行するSQLはユーザが指定することができます。

どの程度の負荷がかかるかはシステム環境に大きく依存しますが、参考値として以下にISE環境でのパフォーマンス結果をご紹介します。

「JDBCドライバーによる検証」を設定しても、ほとんど影響がない程度の負荷であると言えます。「SQL照会による検証」を設定すると、「JDBCドライバーによる検証」より負荷がかかります。(パフォーマンスを保証する訳ではございません。あくまでも参考値です。また、下記はWAS V7での検証結果です。)

環境:WASV7.0.0.1@AIXV6.1 / DB2 V9.5@AIXV5.3 (WASとDB2は別筐体)

同時実行数:1

JDBCドライバーによる検証	5～10ミリ秒の平均レスポンスの劣化
SQL照会による検証	15～25ミリ秒の平均レスポンスの劣化

同時実行数:10

JDBCドライバーによる検証	10～20ミリ秒の平均レスポンスの劣化 2～3%のCPU使用率の増加(WASサーバー)
SQL照会による検証	40～50ミリ秒の平均レスポンスの劣化 6～7%のCPU使用率の増加(WASサーバー)

WASV7以降は非推奨となりましたが、上記パフォーマンス結果が許容されるのであれば、SQL照会による検証もJDBC4.0がサポートされていない環境ではひとつの対応策になるかと思えます。その際、接続検証にて実行するSQL文は、デフォルトのselect current sqldid from sysibm.sysdumy1等、同時に実行してもロックを取得しないテーブルへのアクセスとして下さい。

Designing Web System Infrastructure with WAS V8.0

## 対策2: コーディングによる対応【JDBC 4.0 vs 3.0】

- **JDBC 4.0 環境における接続検証**
  - ◆ 任意のタイミングでisValid メソッドによる接続オブジェクトの検証を行う

★接続を取得した後に接続障害が発生した場合にも、SQL実行前にエラーを検出できる

- **JDBC 3.0 環境における接続失効時対応**
  - ◆ アプリケーションが失効した接続オブジェクトを取得した場合には、実際のSQLを実行した後のタイミングでStaleConnectionExceptionが発生する
    - ▶ StaleConnectionExceptionをキャッチした後のエラー・ハンドリングをアプリケーションの中で記述しておく(リトライなど)

★接続障害発生時のエラー・ハンドリングをコーディングできるため、JDBC4.0環境でも併用することがおすすめ

© 2012 IBM Corporation 41

つづいて、アプリケーション・コーディングにて接続が有効であるかを確認する例を、JDBC 4.0 (isValid)とJDBC3.0(SQL検証)とでご紹介します。

アプリケーション実装とは、ユーザー・アプリケーションにて任意の箇所で、接続オブジェクトの有効性を確認することを指します。

JDBC4.0対応のJDBCドライバが利用できる環境であれば、getConnection実行時以外にも任意のタイミングで、随時 isValidによる接続の検証を行うことができます。

例えばexecuteUpdateの実行直前のタイミングでisValidによる接続の有効性の確認を行うことで、getConnectionの後に障害が発生した場合には一定回数(指定可能)自動再接続を行うようにするなど、Query実行前の対処を柔軟に行うことができるようになります。

JDBC 3.0対応のJDBCドライバを利用する環境を想定する場合、実際のSQLを発行するまで、接続オブジェクトの有効性を確認できないという課題があります。とはいえ、接続障害を示す例外クラスをキャッチして、その中でエラー・ハンドリング処理を記述しておくことで、DBサーバーやネットワークなどの障害に備えることがあるため、JDBC4.0対応ドライバを使う場合も引き続き、こちらのException補填のコーディングを実施いただくことをお勧めします。

Designing Web System Infrastructure with WAS V8.0

### (参考) 設定方法

詳細は、巻末資料「DB接続設計-参考-」をご参照下さい。

- **管理コンソール**: getConnection時の接続検証の実施有無を設定
  - ◆ デフォルトは接続検証を行わない設定
  - ◆ 接続取得時に、自動的に接続検証を行う設定にしたい場合には、管理コンソールにて接続検証を有効にする
- **アプリケーションによる実装**

アプリケーション・コーディング

```

                Connection conn = ds.getConnection();
                if (!conn.isValid(10))
                    // リトライ処理
            }
                Statement s = conn.createStatement();
                ds.executeUpdate(sql);
                ds.close();
            
```

任意の箇所が必要な処理を記述
- **管理コンソール(接続取得時の検証設定)とアプリケーションの違い**

	管理コンソール(接続取得時の検証設定)	アプリケーション
コーディング	必要なし	必要
設定範囲	データ・ソース	アプリケーション
新規接続	再試行回数 x 再試行間隔にて自動再接続を試みる	再試行回数 x 再試行間隔にて自動再接続を試みる
既存プール接続	接続オブジェクトが有効でない場合、別の新しい接続を要求し検証する。バックグラウンドのスレッドにて、DB復旧まで再試行間隔で検証する。	再試行回数 x 再試行間隔にて自動再接続を試みる

管理コンソール設定

接続検証プロパティ

新規接続の検証 デフォルトOFF

再試行回数: 100

再試行間隔: 3 秒間

既存プール接続の検証

再試行回数: 0 秒間

検証オプション

JDBCドライバによる検証

タイムアウト: 秒間

SQL監査による検証 (V7.7では非推奨)

照会: (SELECT CURRENT SQLID FROM SYSIBM.SYSDUMMY1)

接続オブジェクトが有効であるかの検証は、アプリケーションの任意の箇所ではisValidを実行するか、もしくは管理コンソールにて設定することができます。

管理コンソールで指定できるのは、アプリケーションからgetConnectionが実行されたとき、接続をアプリケーションに渡す前にその接続が有効であるか検証を行うかどうかです。

管理コンソールの図で示すように接続検証の設定については、新規接続に対するものと既存プール接続に対するパラメータが別に存在しています。つまり、新規接続を取得するときの接続障害と、既存プール接続を再利用しようとしたときの接続障害について、WASによる自動再接続の機能が異なります。

アプリケーションからの接続要求に対して、新しく接続オブジェクトが作成される状況では、接続検証が検証が失敗、つまりDBサーバとの接続が確立できなかった場合、一定の間隔・回数で接続の再試行が行われます。

既存プール情報の検証では、接続オブジェクトが有効ではない場合、ConnectionManagerが該当の接続オブジェクトをバージし、別の新しい接続を要求します。この接続要求で接続オブジェクトが取得できた場合、実際のSQL文が実行されます。この接続要求で接続オブジェクトが取得できない場合、クライアントに例外が返ります。そして、Connection Managerがバックグラウンドのスレッドを起動し、DBが復旧するまで再試行間隔で接続オブジェクトの有効性を検証します。

OracleなどのDB2以外のデータベースは、データ・ソースのカスタム・プロパティにて設定して下さい。

validateNewConnectionTimeout : 接続取得時のJDBC ドライバによる検証”が可能となる

validateNewConnection : 新規接続の検証

validateNewConnectionRetryCount : 再試行回数

validateNewConnectionRetryInterval : 再試行間隔

validateNewConnectionTimeout : 既存プール接続の検証におけるタイムアウト値

Designing Web System Infrastructure with WAS V8.0

### 接続検証機能 (isValid) の考慮点①

- コールド・スタンバイ(HA)環境における”待ち時間”
  - ◆ アクティブ機からスタンバイ機へのテークオーバーが完了するまでに、通常数十秒程度の時間がかかる
  - ◆ WAS側の自動再接続までの再試行回数・時間はテークオーバーにかかる時間までDB側を待つように設定する
    - ▶ 引継ぎ後のスタンバイ機に自動接続されれば、ユーザーにはエラーが戻らない

**★ポイント:**  
**DBがコールドスタンバイ構成の場合、Takeoverにかかる時間より長く設定する。(ただし長すぎない時間とする)**

© 2012 IBM Corporation 43

データベース・サーバーがコールド・スタンバイ構成となっている場合には、有事の際に数十秒程度の時間がかかります。

スタンバイ側のデータベースが接続を受け付けられる状態となるタイミングまで自動再接続のリトライが行われるよう、再試行回数と時間を調整してください。

Designing Web System Infrastructure with WAS V8.0

## 接続検証機能 (isValid) の考慮点②

- getConnection後にDB障害が起きたら？
  - ◆ 接続検証は接続取得時に接続が失効していない事を確認する機能
  - ◆ 接続取得が完了した後にDB障害、NW障害が発生した場合には、その障害は検知されず、クライアント・アプリケーションには StaleConnectionException が返される
  - ◆ JDBC 4.0(isValid)、JDBC 3.0(SQL検証)いずれにも共通する考慮点

**★ポイント:**  
 接続エラーの発生確率を極力少なくするには、アプリケーションにてエラー (StaleConnectionException) をキャッチするよう実装することがおすすめ

© 2012 IBM Corporation 44

管理コンソールによる接続検証機能を利用すると、getConnectionメソッドによる接続オブジェクトの取得時点の接続失効に対処することができますが、その後のタイミングでDBサーバ障害、ネットワーク障害などが発生し接続オブジェクトが失効した場合、実際のSQLが発行されるタイミングにて StaleConnectionExcpetion がスローされアプリケーションに通知されることとなります。

よって、getConnection後の接続の失効に対応するには、アプリケーションにて StaleConnectionException をキャッチし、リトライするなどの対応が必要となります。

Designing Web System Infrastructure with WAS V8.0

## まとめ: 失効した接続オブジェクトの対応

- isValid() による接続検証の実施を推奨
  - ◆ getConnection() 時に接続の有効性を確認したい場合にはコーディング不要(管理コンソール設定のみで実施可能)
  - ◆ アプリケーション・コーディングにて任意のタイミングで接続が有効であるか検証することも可能
    - バックグラウンドのスレッドにて、接続オブジェクトの有効性を検証する
- 設計のポイント！
  - ◆ 自動再接続の再試行回数・実行間隔は、DBサーバーのテークオーバーに必要な時間を考慮して設定する
  - ◆ アプリケーションでExceptionをcatchする実装は必要
- 【補足】運用による手動対応
  - WAS再起動
  - 管理コンソールによる接続オブジェクトの破棄
  - wsadminコマンドによる接続オブジェクトの破棄

© 2012 IBM Corporation 45

失効した接続オブジェクトの選択/設計指針をまとめます。

StaleConnectionException発生を自動で検知し対応すべきWASV8のシステムでは、アプリケーション実装、もしくは管理コンソールの接続検証プロパティ(JDBCドライバーによる検証)を設定して下さい。接続検証プロパティを設定してもほとんど影響がない程度の負荷であると言えます。また、バックグラウンドのスレッドにて、接続オブジェクトの有効性を検証したい場合には、アプリケーション実装ではなく、管理コンソールを選択して下さい。さらに、getConnection()後の障害に対応するには、今までと同様にStaleConnectionException発生をcatchするロジックを実装して下さい。

また、手動での対応が許可されるシステムの場合には、接続オブジェクト失効後に、以下のどちらかを実施することを運用に組み込むことをご検討下さい。

- ・WAS再起動を行う
- ・管理コンソール、もしくはwsadminコマンドにて全ての接続オブジェクトを破棄する

Designing Web System Infrastructure with WAS V8.0

## DB2 Client Reroute (IPの異なるDBサーバへの切替)

- 接続先DBに障害が発生した場合、Client側で接続先を切り替える機能

- 設定方法
  - ◆ 管理コンソールより、JDBC -> データ・ソース -> WebSphere Application Server データ・ソース・プロパティを選択する
  - ◆ V7~
    - 設定方法の簡便化
    - クライアント転送サーバー・リストJNDI名
      - 転送情報(プライマリーDB、代替DB)をnamespaceにバインド可能
- 前提条件
  - DB2 for z/OS Version 9.1 or later / DB2 Version 9.5 or later
  - DPF / DPRORP / PowerHA / HADR
  - データ・ソースの設定
- 設計指針
  - ◆ プライマリーDBと代替DBのIP/ポート番号が異なる場合に利用
  - ◆ ユーザーへの影響を最小化するには、アプリケーション側での StaleConnectionExceptionなどの例外ハンドリングを組み合わせる

DB2 自動クライアント転送オプション

クライアント転送の再試行間隔: 10 秒間

クライアント転送の最大再試行回数: 5 回数

代替サーバー名: 192.168.0.2

代替ポート番号: 60001

クライアント転送サーバー・リスト JNDI 名:

JNDI からのアンバインド・クライアント転送リスト

© 2012 IBM Corporation 46

DB2 Client Rerouteとは、接続先DBに障害が発生した場合、Client側で接続先を自動的に切り替える機能です。

Client Reroute機能を使用することで、アプリケーションは切断されてしまったDBサーバーへの再接続処理を行う必要がなくなります。この機能は、HADR/Replicationを利用する環境等、PrimaryDBサーバーと代替DBサーバーのIPアドレスが異なる場合に有用です。

WAS V7から、管理コンソールでDB2の代替サーバー情報を設定することが可能になりました。WAS V6.xの環境では、再試行間隔と再試行回数をデータ・ソースのカスタムプロパティに、それ以外はType4の場合はアプリケーション・コーディングにて、Type2の場合にはDBサーバー側でUPDATE ALTERNATE SERVER FOR DATABASEコマンドにて、代替サーバー情報を登録していました。さらに、オプションで、Type4ドライバーでのみ、転送情報(プライマリーDB情報、代替DB情報)をnamespaceにバインドすることも可能です。

DB2 Client Reroute機能を設定した際の考慮点は以下になります。これは、IPアドレスの付け替えを伴うHA構成(DBサーバー)とするか、IPアドレスの付け替えないHADR(DBサーバー) + DB2 Client Reroute構成とするかの考慮点ともなります。

- ・代替DBへ再接続された場合、PrimaryDBで実行中であつたトランザクションは自動的にロールバックされる。この場合、トランザクションの再実行はアプリケーションでハンドリングする必要がある。
- ・Primary DB→代替DBの双方に対して交互に自動再接続処理が実行される
- ・1度代替サーバーへ接続されると、以降はすべて代替サーバーへ接続する

・WASV8.0 Information Center - 「DB2 データベースを使用するアプリケーションのクライアント・リルート構成」

[http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tadat\\_clientreroute.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tadat_clientreroute.html)

・DB2 Workshop資料 - 「クライアント・リルートとWAS連携の考慮点」

[http://www.ibm.com/jp/domino01/mkt/dminfo.nsf/499721c3388537bd49256b1a001aab28/4925722f004efcee492570dd000db513/\\$FILE/CR+WAS.pdf](http://www.ibm.com/jp/domino01/mkt/dminfo.nsf/499721c3388537bd49256b1a001aab28/4925722f004efcee492570dd000db513/$FILE/CR+WAS.pdf)

Designing Web System Infrastructure with WAS V8.0

---

## 2. WAS-DB2接続設計

- 2-1 JDBCドライバー設計
- 2-2 データ・ソース設計
  - a. セキュリティー
  - b. 接続プール
  - c. データ・ソース・プロパティー
  - d. DB2 pureScaleへの接続
- 2-3 アプリケーション設計
- 2-4 パフォーマンス / 問題判別
- 2-5 Hints & Tips

© 2012 IBM Corporation 47

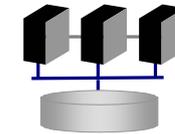
続きまして、WASとDB2 pureScaleとの接続について、ご説明致します。

Designing Web System Infrastructure with WAS V8.0

## WAS - DB2 pureScaleへの接続

- DB2 pureScale
  - ◆ クライアントからの多数のデータアクセスを複数のDBサーバーで並行して処理することで同時実行性を高めたアーキテクチャーのデータベース・サーバー群
  - ◆ WAS V7.0からサポート
- WAS-DB2 pureScale サポートされる組み合わせ
  - ◆ WAS V7.0 - DB2 pureScale V9.8
    - JDBCドライバー ( IBM Data Server Driver for JDBC and SQLJ ) はバージョン 3.57 or 4.7 以上がサポート要件
    - WASへのiFix適用が必要 (APAR IC64352)
  - ◆ WAS V8.0 - DB2 pureScale V9.8
    - JDBCドライバー (同上) はバージョン 4.8.87 以上がサポート要件

ロード・バランス型  
クラスタリング構成



アクティブ-アクティブ型 シェアード・ディスク型 製品例: DB2 pureScale, OracleRAC
構成サーバー全てでDBサーバー・プロセスが稼動し、クライアントからの要求を処理する
接続先となるDBサーバー全てのホスト名 (IP)、Port をWASデータ・ソースに登録しておく構成が一般的
全サーバーに障害が起きない限りDB接続は可能。但し単一サーバー障害時は、ログからの回復が完了するまで該当データへのアクセスが制限される。

© 2012 IBM Corporation
48

DB2 pureScaleは、IBMが提供するシェアード・ディスク型のアクティブ-アクティブ構成のデータベース・サーバーです。

WASでは、V7.0からpureScaleへの接続をサポートしています。

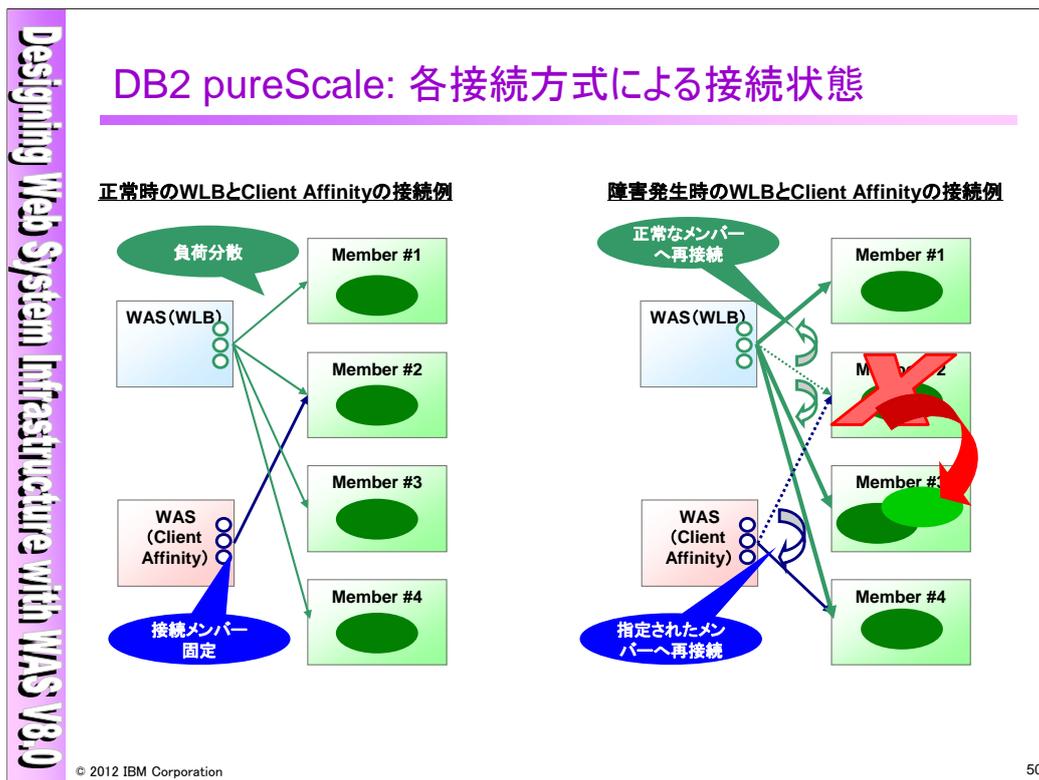
## DB2 pureScale: 接続方式の検討

- 接続方式を検討する
  - ◆ 接続方式に応じて、データ・ソースを作成する必要がある

	負荷分散を行いたい場合 (WLB)	接続先を固定したい場合 (クライアント・アフィニティ)
接続先	各DB2メンバーの負荷に応じて自動的に接続先を変更	指定したDB2メンバーに接続(指定方法は後述)
障害発生時の接続先の変更	生存している他のDB2メンバーに再接続	指定したDB2メンバー(代替サーバー)に再接続
障害発生時のアプリケーションへの影響	障害発生したメンバーに接続していたアプリのみ、再実行が完了されるまで遅延する。特に更新しかかり中の接続の場合、再接続完了の通知が返される(JCCの場合、“-4498”)。またシームレス・フェールオーバーを使用する場合、更新をしかける前および参照系のアプリの場合は、エラーは返されない。	

- 接続タイムアウト値を設定する
  - ◆ システムの接続タイムアウト値とは別に設定することができる

DB2 pureScaleは、純粹に負荷分散を行うWLBのモードと、接続先を固定し有事の際だけ指定した代替サーバーに再接続するクライアント・アフィニティのモードとを利用することができます。



WLBとClient Affinityそれぞれの接続と障害時の再接続について説明した図です。

左側が正常時、右側が障害発生時の状況を示します。

図の上側、深緑の○・矢印・吹き出しで描かれるのがWLBの接続、図の下側、青い○・矢印・吹き出しで描かれるのがClient Affinityの接続です。

WLBにてpureScaleに接続しているWASは、Member #1-4までのDB2サーバのいずれかに接続されます。(負荷分散)

この図での例のように、Member#2に障害が起きた場合、Member#2に接続していたWASは、次回の接続時には残りの生存サーバであるMember#1,3,4 いずれかに接続されます。また、Member#2上で動作していたDB2のプロセスはMember#3上で引き続き動作しつづけます。

Client Affinityで接続している場合、WASはこの図ではMember#2にのみ接続されます。

ただしMember#2に障害が発生した場合、指定されたメンバー、この図ではMember#4へ再接続されるようになります。

Designing Web System Infrastructure with WAS V8.0

## WAS - DB2 pureScale 【WLB or クライアント・アフィニティー】

- 一般的な負荷分散用途としては、WLB にて接続する
  - ◆ 複数台のノードで要求を分散することでスケーラビリティを高める
  - ◆ 通常のOLTP=所要時間の短い、ResultSetの小さいSQL であればWLBを選択
- OLTPシステム (Webシステム) でクライアント・アフィニティーを使うケースとは？
  - ◆ 大量検索や、レポート系の処理などDBサーバーのCapacityを占領してしまうような処理について、特定のDBサーバー上で処理をさせるよう固定することで、他のOLTP系の処理を行うクライアントへの影響を最小化するケースなど。
- 同一のWASノードからWLB、クライアント・アフィニティーそれぞれのモードで pureScaleへ接続する場合は、WASデータ・ソースを分けて定義する

© 2012 IBM Corporation

51

WASから接続するケースでは、一般的な負荷分散を用途とするケースが多いと考えられ、この場合はWLBにて接続するようWAS側を構成していただければと思います。複数台のDB2サーバーでSQL要求を並列処理できる構成であることから、特に、1回あたりの取得データが小さくユーザー数が多いケースで性能向上に寄与する構成といえます。

ただし、1回あたりの取得データが大きくDBサーバ上の処理時間やCPU使用率が有意に大きくなるものが想定される処理を行う場合、

たとえば大量検索やレポート系の処理がある場合には、ClientAffinityのモードで特定のDBサーバー上に接続されるようにし、他のクライアント(⇒他のpureScale Memberに接続する)に影響が及ばないようにする構成が、全体の性能要件を満たす上で有効と考えられます。

なお、同一のWASからWLB、Client Affinityの両方のモードで接続することもできますが、WLB接続用のデータ・ソースとClient Affinity接続用のデータ・ソースとを分けて定義しておく必要があります。

Designing Web System Infrastructure with WAS V8.0

## DB2 pureScale: WLB 【WASでの設定例】

- 1. 通常のデータソースの作成(1データソース)
  - ▶ 接続先serverNameとして定義するDBサーバ(pureScaleメンバー)は1つでOK
  - ▶ 最初に接続したDBサーバ(serverName)から、接続先サーバー・リストが送信される
- 2. WLB接続を行うためのデータソース・カスタム・プロパティの設定
  - ◆ enableSysplexWLB
    - ▶ WLBを有効にする場合、この値を”true”に設定する
  - ◆ seamlessFailover
    - ▶ WLBを有効にすると、自動的に有効になる
    - ▶ 明示的に有効にする場合は、”1”を設定する
  - ◆ 管理コンソールでの設定例:
 

<input type="checkbox"/>	<a href="#">enableSysplexWLB</a>	true
<input type="checkbox"/>	<a href="#">enableSeamlessFailover</a>	1
- 3. purgePolicyはFailingConnectionOnlyに設定
  - ◆ 障害が発生したメンバーに接続していた分のみを、正常な他のメンバーへリルートできるようにするため

© 2012 IBM Corporation 52

WLBモードでDB2 pureScaleを利用する場合には、WASデータソースのカスタム・プロパティとして下記を設定します。

- enableSysplexWLB: true
- seamlessFailover: 1(無指定時には自動的に1=有効 となる)

OracleRACを利用するときと同様、接続が失効した場合のPooled Connection 廃棄のポリシー(purgePolicy)としては、

FailingConnectionOnlyとします。

EntirePool とすると、他の生きているノードにつながっている、再利用可能な接続まで一緒に破棄されてしまうためです。

Designing Web System Infrastructure with WAS V8.0

### (参考)WLBでのサーバー・リストの取得の流れ

- 1. WASデータ・ソースに定義されたpureScaleメンバーに接続 (ここではMember1)
- 2. Member1からWASへサーバー・リストを送付
- 3. WASからは、リストに掲載されたメンバーへ順次接続

- 【Tips】初回接続先につながらない場合への対策は？
  - ◆ データ・ソースのカスタム・プロパティ「clientRerouteAlternateServername」に、接続先候補のサーバー名を複数リストしておくことができる

© 2012 IBM Corporation 53

DB2 pureScale (WLB) では、接続可能なサーバー名と優先度を含んだサーバー・リストが配布されることによって、接続先となるべきDBサーバーの情報をクライアント (=WAS) に通知します。  
 WASからは、サーバー・リストの情報を元に、各Memberへの接続を行います。

Designing Web System Infrastructure with WAS V8.0

## DB2 pureScale: クライアント・アフィニティー【WASでの設定例】

- 1. 通常のデータ・ソースの作成(1データ・ソース)
  - ▶ 接続先serverNameに定義するDBサーバ(pureScaleメンバー)は1つでOK
  - ▶ 優先サーバ、代替サーバの設定はclientRerouteAlternateServerNameにて指定する
- 2. クライアント・アフィニティー接続を行うためのデータ・ソース・カスタム・プロパティの設定

<input type="checkbox"/>	<a href="#">enableClientAffinitiesList</a>	1
<input type="checkbox"/>	<a href="#">clientRerouteAlternateServerName</a>	pub31,pub51
<input type="checkbox"/>	<a href="#">clientRerouteAlternatePortNumber</a>	60010,60010
<input type="checkbox"/>	<a href="#">affinityFailbackInterval</a>	300
<input type="checkbox"/>	<a href="#">enableSeamlessFailover</a>	1

接続の優先順にサーバー名とポート番号をカンマ区切りで指定する。  
この場合は、pub31が優先的に接続される。

pub31が復旧した後に、pub31にフェールバックすることができる。

© 2012 IBM Corporation 54

Client Affinityを想定してDB2 pureScaleメンバーに接続するには、通常のデータ・ソースを作成し、データ・ソース・カスタム・プロパティとしてclientRerouteAlternateServerName等を設定します。

clientRerouteAlternateServerNameには、優先順にサーバー名を記述します。また、優先サーバが障害から回復した後にフェールバックさせるかどうかなどの設定も、同様にカスタム・プロパティで設定することができます。

## DB2 pureScale参考情報

- DB2 pureScale
  - ◆ <http://www.ibm.com/software/jp/data/db2/v9/purescale/>
- DB2 pureScale 製品マニュアル
  - ◆ <http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/index.jsp>
- 実践 DB2 pureScaleテクニカル・セミナー
  - ◆ [http://www.ibm.com/developerworks/jp/data/library/dataserver/purescale\\_seminar.html](http://www.ibm.com/developerworks/jp/data/library/dataserver/purescale_seminar.html)



DB2 pureScaleの情報リンクです。

Designing Web System Infrastructure with WAS V8.0

### まとめ: データ・ソース設計 ～データ・ソース・プロパティー

項目	選択肢	設計指針
Prepared Statement Cache (参考資料)	<ul style="list-style-type: none"> <li>・WAS</li> <li>Prepared Statement Cache</li> <li>・DB2</li> <li>db2agentのヒープサイズ</li> </ul>	<ul style="list-style-type: none"> <li>・WAS</li> <li>発行されるSQL数を設定して下さい。(1接続単位)</li> <li>・DB2</li> <li>WAS側の設定を考慮して、CLIPKG (パッケージ数) とdb2agentのヒープサイズ (applheapsz)をチューニングして下さい。</li> </ul>
失効した接続オブジェクト	<ul style="list-style-type: none"> <li>&lt;製品機能/実装による自動対応&gt;</li> <li>・接続検証プロパティー</li> <li>・アプリケーション実装</li> <li>&lt;運用による手動対応&gt;</li> <li>・WAS再起動</li> <li>・管理コンソール/wsadminコマンド</li> </ul>	<ul style="list-style-type: none"> <li>JDBCドライバーによる接続検証 (JDBC4.0新機能) の活用を推奨します。</li> <li>StaleConnetionExceptionをcatchするコーディングによる接続取得後の障害対応の実施もあわせて検討して下さい。</li> </ul>
DB2 Client Reroute	-	プライマリーDBと代替DBのホスト名/ポート番号が異なる場合、代替DBへフェールオーバーされても、DBへの接続処理を実施せず、トランザクションを実行したい場合に検討して下さい。
拡張DB2データ・ソース(参考資料)	非コア・カスタム・プロパティー currentSchema、traceLevel 等	アプリケーション (EAR) 毎に異なるカスタム・プロパティー値を設定し、かつ接続プールを共用したい場合に検討して下さい。
pureScale	-	用途が一般的な負荷分散であれば、WLBモードを選択します。

56

データソース設計のデータ・ソース・プロパティーについて、その設計指針をまとめます。

Designing Web System Infrastructure with WAS V8.0

---

## 2. WAS-DB2接続設計

- 2-1 JDBCドライバー設計
- 2-2 データソース設計
  - a. セキュリティー
  - b. 接続プール
  - c. データソース・プロパティー
  - d. DB2 pureScaleへの接続
- 2-3 **アプリケーション設計**
- 2-4 パフォーマンス / 問題判別
- 2-5 Hints & Tips

© 2012 IBM Corporation 57

続きまして、アプリケーション設計についてご説明致します。

アプリケーションコードにて記述するJNDI名について、アプリケーションのメタデータであるデプロイメント・ディスクリプターにて指定できる内容について説明します。

また、DB2の最新版である V9.7 では、デフォルトの分離レベルが変更されていますのでこちらについてもご紹介致します。

Designing Web System Infrastructure with WAS V8.0

## 分離レベル

- 分離レベルとは？
  - ◆ 複数の処理が同時に実行された場合、他の処理からの影響度合いを定義したレベルのこと

JDBC		DB2	
TRANSACTION_SERIALIZABLE	S	RR	Repeatable read
TRANSACTION_REPEATABLE_READ	RR	RS	Read stability
TRANSACTION_READ_COMMITTED	RC	CS	Cursor stability
TRANSACTION_READ_UNCOMMITTED	RU	UR	Uncommitted read

- 分離レベルはどこで設定する？
  - ◆ データベース全体、アプリケーション(EAR)、セッション(接続)、ステートメントの単位で設定することができる
    - より狭い範囲で設定したものは、広い範囲の分離レベル設定を上書きする
    - 但しWASではステートメントにて都度変更する方法は非推奨。  
接続プールに異なる分離レベルの設定された接続が混在し、アプリケーションが想定外の挙動を示す可能性があるため。
- <DB2 V9.7 New>デフォルトの分離レベルの動作の変更
  - ◆ DB2 V9.7 では、READ COMMITTED(CS)の動作が変更された(⇒次頁)
  - ◆ WASの場合、デフォルトの分離レベルはREPEATABLE READ(WAS V5～)
  - ◆ 性能と平行性の観点から、基本的にREAD COMMITTED(CS)がおすすめ

© 2012 IBM Corporation 58

分離レベルとは、DBMS上でのデータ・アクセス処理(トランザクション)が複数同時に行われた場合に、どれほどの一貫性、正確性で実行するかを4段階で定義したものです。

DB2 V9.7 では、デフォルトの分離レベルであるCS(Cursor Stability)の挙動が変わりました。この新しいCSの振る舞いについて、「Currently Committed」と呼ばれることがありますが、分離レベルの名前としては「CS」のみであり、変更はありません。

DB2 V9.5以前のCSとV9.7のCS(CC)では、見るタイミングでロック待ちになるかならないかの違いがあります。

V9.5以前では、更新が始まったデータは更新が完了するまでロック解放待ちとなりましたが、V9.7では、ロック待ちをせず、直前のタイミングでコミットされたデータが即時に戻されるように仕様が変更されました。

とはいえ、WAS上にデプロイされるアプリケーションでは、より厳密な分離レベルであるRepeatable Readが適用されるためこちらの仕様変更の影響を受けませんが、明示的にReadCommitted(CS)を指定する場合には、上記挙動が変わりますので留意ください。

Designing Web System Infrastructure with WAS V8.0

## 新しい Cursor Stability モードの動作

- DB2 V9.7独自の読取り一貫性 (Currently Committed)
  - ◆ DB2の分離レベルの名称は、CSのまま
  - ◆ **未コミットの更新があってもロック待機をしない**
    - ▶ 参照処理は、更新処理に伴うロックの開放を待たず、更新前のデータ(コミット済みの最新データ)をログ(ログ・バッファ)から読む
    - ▶ 常に最新のコミット済みデータを読む
  - ◆ Oracleのデフォルト分離レベルに近い動作を示すようになった
  - ◆ DB2 V9.5以前のCSの動作での稼動も可能(DB構成/パラメータ: CUR\_COMMIT)

DB2: 参照処理はロック待機せず、かつコミット済みの最新データを読む

© 2012 IBM Corporation 59

DB2 V9.7では、デフォルトの分離レベルであるCursor Stability(CS) の動作がこれまでと変更されています。

TRN1がロックしているデータをTRN2が読み取ろうとしたとき、これまでのDB2ではロック待機を行っていましたが

DB2 V9.7 ではロック待機をせず、SELECTした時点でコミット済みとなっている最新の値がトランザクションログ(ログ・バッファ)から読み出され、TRN2に渡されます。

常に最新のコミット済みデータを読む動作となっており、旧来のDB2のCSに比べるとOracleのデフォルト分離レベルに近い動作となりました。

とはいえ、厳密にはOracleデフォルト分離レベルの動作とは異なります。(⇒次ページ)

### ■DB2デザイン・ガイド: ロックの基礎【V9.7対応】

[http://www.ibm.com/developerworks/jp/data/products/db2/design-guide/lock\\_v97.html](http://www.ibm.com/developerworks/jp/data/products/db2/design-guide/lock_v97.html)

Designing Web System Infrastructure with WAS V8.0

## 分離レベル: DB2 と OracleDB との違い

- DB2とOracleのデフォルトの分離レベルは、詳細には異なります
  - ◆ DB2 : データ読み込み時点のコミット済み最新データを読む
  - ◆ Oracle: 該当セッション開始時点のコミット済み最新データを読む

**【Oracle の場合の読取り一貫性(READ COMMITTED)】**

Oracle : 読んだデータが最新とは限らないが、参照処理はロック待機しない

© 2012 IBM Corporation 60

<DB2とOracleのデフォルトの分離レベルの違い>

DB2では、データ読み込み時点のコミット済み最新データを読みとります。

Oracleは、該当セッション開始時点のコミット済み最新データを読みとります。

このページの図では、Oracleのデフォルト分離レベルReadCommittedの動作を説明しています。  
(DB2の挙動は前ページを参照)

<図の説明>

OracleではSCN(システム変更番号)と呼ばれる、論理的な内部タイムスタンプを用いて時系列にデータ更新イベントを監視・管理しています。

図では、TRN1とTRN2という2つのトランザクションが同じ時間帯に実行されていたものと想定しています。

先に開始したのは、右側のTRN2のセッション(SCN=100)ですが、他の行データ読み取り処理を行ううちに、次に読み込む予定であったC/Dのデータに対し、より遅く開始されたTRN1(SCN=110)が先に更新処理を行ったとします。

より新しいセッション(SCN=110)による更新処理でデータが変更されている場合、SCN=110のセッションによりコミットされた最新のデータではなく、SCN=100の時点でコミットされている最新データ(SCN=29時点の変更が反映されている)の値をUNDO領域から読み取るという制御が行われています。

Designing Web System Infrastructure with WAS V8.0

## WASデフォルトの分離レベル

- 各DBMSにWASからアクセスする際のデフォルト分離レベルは下表の通りです。

DBMS	分離レベル
Cloudscape	REPEATABLE READ
DB2	REPEATABLE READ
DB2/AS400	REPEATABLE READ
Informix	REPEATABLE READ
Microsoft SQL Server	REPEATABLE READ
Oracle	READ COMMITTED
Sybase	REPEATABLE READ
Unsupported databases	READ COMMITTED

© 2012 IBM Corporation 61

出典:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/welcome\\_ndmp.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/welcome_ndmp.html)

Designing Web System Infrastructure with WAS V8.0

### まとめ:アプリケーション設計

項目	選択肢	設計指針
分離レベル	<ul style="list-style-type: none"> <li>・&lt;CMP Entity Beanの場合&gt;</li> <li>アクセス・intent選択による自動指定</li> <li>&lt;上記以外のWebModuleやEJB&gt;</li> <li>・ibm-web-ext.xml</li> <li>・setTransactionIsolation()</li> <li>・SQL文のwith句</li> <li>・データ・ソース・カスタム・プロパティ</li> </ul>	<p>アプリケーションの設定として、リソース参照 (ibm-web-ext.xml)にて指定します。下記いずれの分離レベルに設定するべきかは業務アプリケーションの実装に依存します。業務チームと連携し、EAR/WARを作成する時点で、適切な分離レベルが指定されるようにして下さい。</p>
JNDI lookup (参考資料)	<ul style="list-style-type: none"> <li>・直接参照</li> <li>・リソース参照</li> </ul>	<ul style="list-style-type: none"> <li>・リソース参照</li> </ul> <p>参照名とJNDI名のマッピングをデプロイメント記述子で行って下さい。</p> <p>アプリケーションからは、参照名をlookupします。</p> <p>@Resourceの場合は、認証方式と共用スコープをアプリケーション・コード内に記述することになりますので、ご注意下さい。</p>
共用スコープ (参考資料)	<ul style="list-style-type: none"> <li>・共用可能接続</li> <li>・共用不可能接続</li> </ul>	<p>パフォーマンス・テストにて接続プールが枯渇していないかを確認して下さい。</p> <p>共用不可能接続の方が、効率よく接続プールを使い回しできる可能性が高いです。</p> <p>共用スコープを変更する場合には、アプリケーションへの影響がないかを確認して下さい。</p>

© 2012 IBM Corporation 62

Designing Web System Infrastructure with WAS V8.0

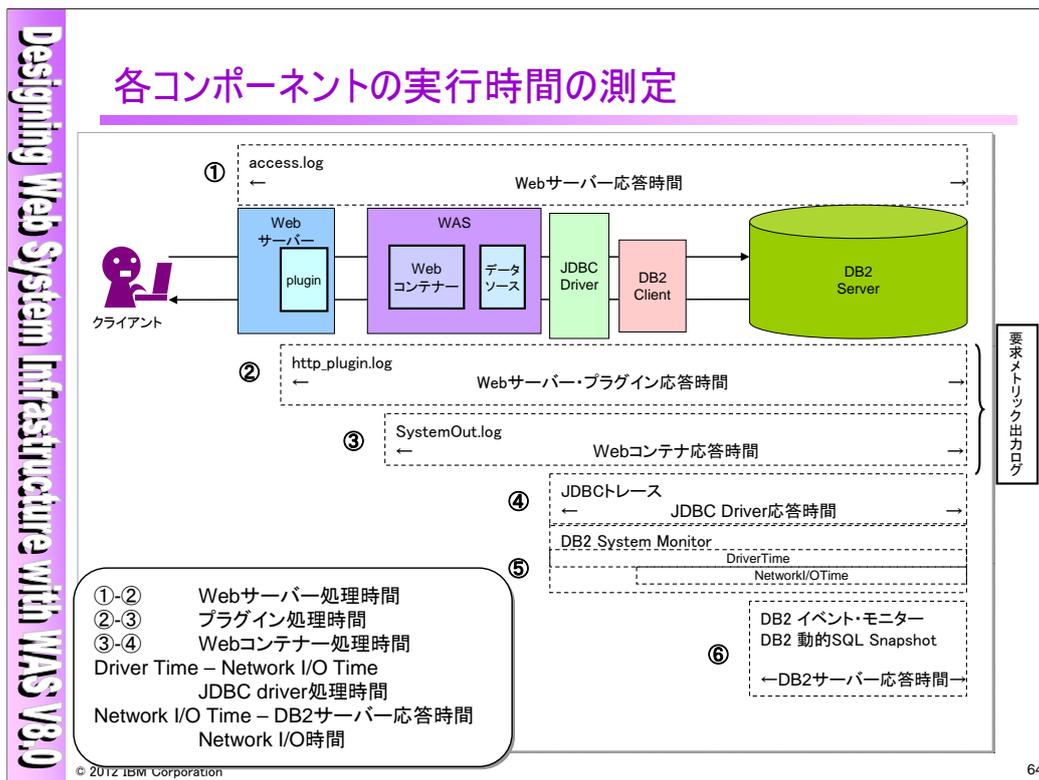
---

## 2. WAS-DB2接続設計

- 2-1 JDBCドライバー設計
- 2-2 データ・ソース設計
  - a. セキュリティー
  - b. 接続プール
  - c. データ・ソース・プロパティ
  - d. DB2 pureScaleへの接続
- 2-3 アプリケーション設計
- 2-4 パフォーマンス / 問題判別
  - 性能監視機能
- 2-5 Hints & Tips

© 2012 IBM Corporation 63

続きまして、パフォーマンス / 問題判別についてご説明致します。



はじめに、IHS / WAS / DB2を使用したシステムにおける、実行時間の測定方法をまとめます。

各コンポーネント毎の実行時間を知るためには各コンポーネントのログを取得する必要がありますが、全てを取得した場合、ログ出力量が膨大になり、リソースを消費してしまいます。また、そのログ出力に一番時間を要してしまうということも考えられます。従いまして、IHSのaccess\_log等で問題箇所を絞り込み、特定のアプリケーションのみ実行した状態で該当コンポーネントのログを取得するという工夫が必要です。

また、動的SQLスナップショットでは、1SQL実行当りの平均実行時間しか分かりません。同じSQLなのに特定の時間帯のみ遅い、特定のアプリケーションから実行されたSQLが遅い等の場合には、イベント・モニターにて実行時間を取得して下さい。

- ①Webサーバー応答時間から②Webサーバー・プラグイン応答時間を引いたものが、**Webサーバー処理時間**になります。
- ②Webサーバー・プラグイン応答時間から③Webコンテナ応答時間を引いたものが、**プラグイン処理時間**になります。
- ③Webコンテナ応答時間から④JDBC Driver応答時間を引いたものが、**Webコンテナ処理時間**になります。
- ④JDBC Driver応答時間から⑥DB2サーバー応答時間を引いたものが、**JDBC Driver処理時間 + Network I/O時間**になります。
- ⑤System Monitorを取得した場合、Driver TimeからNetwork I/O Timeを引いたものが、**JDBC driver処理時間**になります。
- ⑤System Monitorを取得した場合、Network I/O Timeから⑥DB2サーバー応答時間を引いたものが、**Network I/O時間**になります。

Designing Web System Infrastructure with WAS V8.0

### 性能監視 ～WAS-DB2の処理時間を把握する～

1. DB2 System Monitor
2. WASTレース設定
3. JDBCトレース設定

クライアント → Web サーバー (plugin) → WAS (Web コンテナ, データソース) → JDBC Driver → (DB2 Client) → DB2 Server

WASTレース  
JDBCトレース  
DB2 System Monitor

JDBCドライバーの設定方法は、巻末資料「DB接続設計-参考-」をご参照下さい。

© 2012 IBM Corporation 65

WAS-DB2間の処理が遅いというときに原因となっている箇所を特定するために利用できるトレース機能についてご紹介します。

■DB2 System Monitor

各コンポーネントでの処理時間の取得

■WASTレースの取得方法

どのSQLやEJB呼び出しが実行されているかの確認

WAS接続プールの利用状況の確認

■JDBCトレースの取得方法

JDBCドライバーとDBサーバ間の通信にかかっている時間や、JDBCドライバ上の処理の流れの確認

詳細は、巻末資料をご参照ください。

Designing Web System Infrastructure with WAS V8.0

---

Hints & Tips

- タイムアウト関連の設定！
- インメモリKVSによるデータアクセス性能向上アプローチ！

## 2. WAS-DB2接続設計

- 2-1 JDBCドライバー設計
- 2-2 データ・ソース設計
  - セキュリティー
  - 接続プール
  - データ・ソース・プロパティー
  - DB2 pureScale接続
- 2-3 アプリケーション設計
- 2-4 パフォーマンス / 問題判別
- 2-5 [Hints & Tips](#)

© 2012 IBM Corporation 66

続きまして、WAS-DB2接続設計におけるHints & Tipsについてご説明致します。

Designing Web System Infrastructure with WAS V8.0

## WAS-DB2接続タイムアウトに関わる設定

- 「タイムアウト」 = 処理時間に一定の上限を設けておくこと
  - ◆ ある処理に時間がかかり過ぎた場合に、処理を中断しエラーを返すことで、ユーザーに処理の失敗を通知する
    - ⇒ クライアントがサーバーからの応答を無限に待たずに済む
- お客様要件にあわせて適切な設定値をご確認ください。
  - ◆ タイムアウトが長すぎると、クライアント(ブラウザ)への応答が長時間戻らない
  - ◆ タイムアウトが短すぎると、必要な処理が常に中断されエラーとなる

タイムアウトの設定については、巻末資料「DB接続設計-参考-」をご参照下さい。

  : WASデータソースのプロパティ  
  : DB2用JDBCドライバのプロパティ  
  : OS設定

67

タイムアウトは、サーバーの高負荷状態やネットワーク障害時などに、クライアントが無限に待つ事態を発生させないために必要な考え方です。

時間がかかりすぎた場合にはいったん処理を中断してクライアントにエラーを返すことで、クライアント側で状況を把握でき、時間がたったらリトライするなど対応を取ることができます。

WASからDB2への接続中におけるタイムアウト時間を決めるための設定項目として、当ページの図に記したようなパラメーターがあります。

各パラメーターについては、巻末資料「DB接続設計 -参考-」の資料に詳しく記載しております。

Designing Web System Infrastructure with WAS V8.0

## KVSによるデータ・アクセス性能向上【WXS活用のススメ】

- JDBC接続の性能が上がらないのが、ロック競合である場合には、サーバーのスケールアップ、スケールアウトなど従来のアプローチでは性能向上が見込めない。
- そこで、特にアクセス競合の激しいデータについて、KVSベースのインメモリー・データ・キャッシュに配置することでRDBMS上のロック競合を回避し高速にデータアクセスを行う方式も有効な選択肢となる。
- 基本的には製品提供APIによるデータ・アクセスを行うようアプリの改修が必要となり敷居が高いが、JPAでアクセスされるDBデータやWAS Sessionオブジェクトであればインメモリーのキャッシュ機能を手軽に利用できることから、これらの技術を使っていて性能向上が課題である場合にはお勧め。

■従来型の  
データ・アクセス

データアクセスが集中することによりデータベースサーバーがボトルネックになる

■WXS経由の  
高速データ・アクセス

一度読み込んだデータをキャッシュすることで2度目以降のデータアクセスを高速化。WASサーバー間でキャッシュ領域を共有・有効活用。

© 2012 IBM Corporation
68

JDBC接続の性能が上がらないのが、ロック競合である場合には、サーバーのスケールアップ、スケールアウトなど従来のアプローチでは性能向上が見込めません。

そこで、特にアクセス競合の激しいデータについてKVSベースのインメモリー・データ・キャッシュに配置することでRDBMS上のロック競合を回避し高速にデータ・アクセスを行う方式が有効であるケースがあります。

KVSを利用する場合にはJDBCではなく、各製品提供APIによるデータアクセスを行うようアプリケーション改修が必要となることや、KVS上ではデータが非正規化された形で保持されるため更新処理実装について要件に基づいた検討が必要となることなど、考慮点は存在します。

ただし、JPAでアクセスされるDBデータやWAS Sessionオブジェクトであればアプリ改修なしでインメモリーのキャッシュ機能を手軽に利用できることから、これらの技術を使っていて性能向上が課題であるケースにはお勧めしたい製品です。



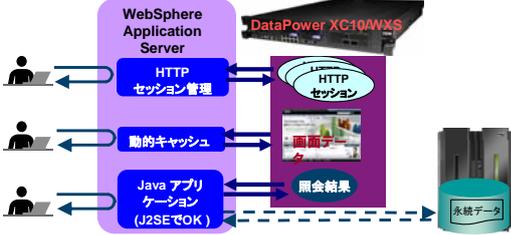
DH1 Format

**【ご参考】 WebSphere eXtreme Scale(WXS: インメモリ・データ・グリッド製品)  
IBM WebSphere DataPower XC10 アプライアンス(キャッシュ専用アプライアンス)**



**WXS / XC10 アプライアンスは、エンタープライズ・アプリケーション基盤の  
キャッシュ層において、Webアプリケーションの高速化と可用性を実現いたします !!**

【実現できる機能】	【期待されるメリット】
<ul style="list-style-type: none"> <li>▪ WebSphere Application Server の HTTPセッションを WXS / DataPower XC10 に保管・共有                             <ul style="list-style-type: none"> <li>▪ 複数のBaseエディション、Expressエディション間でセッションの共有が可能</li> <li>▪ NDエディションにおいても、複数のセルに跨ったアプリケーション・サーバー間で共有が可能</li> </ul> </li> <li>▪ WebSphere Application Server の動的コンテンツのキャッシュ(DynaCache)を WXS / XC10 に保管                             <ul style="list-style-type: none"> <li>▪ 同様のリクエストに対しては、キャッシュを利用するため、DBアクセスやビジネス・ロジックの実行が不要</li> <li>▪ キャッシュによりユーザーへのレスポンスが向上</li> </ul> </li> <li>▪ 大容量(240GB)のキー・バリュー・ストアとして利用                             <ul style="list-style-type: none"> <li>▪ クライアントはJava SE 1.4以上で稼動し、WebSphere Application Server 以外もサポート</li> <li>▪ クライアントは無償で提供</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>▪ セッション共有による可用性向上と運用負荷削減</li> <li>▪ Webアプリケーションのレスポンスの高速化</li> <li>▪ バックエンドのDBやホストシステムへの負荷を削減</li> <li>▪ サーバー台数削減によるHW/ライセンス/運用コスト削減</li> <li>▪ サービス開始までの時間短縮</li> </ul>



69 © 2010 IBM Corporation

Designing Web System Infrastructure with WAS V8.0

### まとめ: Hints & Tips

項目	選択肢	設計指針
タイムアウト設計	<ul style="list-style-type: none"> <li>•setQueryTimeout</li> <li>•WAS接続タイムアウト/滞留タイマー</li> <li>•JDBCドライバー・プロパティ</li> <li>•TCP/IP Timeout設定</li> </ul>	<p>以下を考慮し、適切な設定値をご検討ください。</p> <ul style="list-style-type: none"> <li>•タイムアウトが長すぎると、クライアント(ブラウザ)への応答が長時間戻らない</li> <li>•タイムアウトが短すぎると、必要な処理が常に中断されエラーとなる</li> </ul>
KVS (WXS/XC10)	<ul style="list-style-type: none"> <li>•RDBデータを WXS 上にキャッシュ</li> <li>•Session を WXS 上にキャッシュ</li> <li>•JPA DataObject を WXS 上にキャッシュ</li> </ul>	RDBデータをWXS上に配置する場合にはアプリケーション改修が必要となるため、キャッシュによる効果が得られるアプリケーション/データ構造であるのかを検討する。Session/JPA Objectであれば、比較的少ない手間でWXSによる性能向上の恩恵が受けられる。
ワークロード管理 (参考資料)	<ul style="list-style-type: none"> <li>•ClientInformationAPI</li> <li>•DB2 ワークロード管理</li> <li>(•AIX CPU/プリフェッチ制御)</li> <li>(•WebSphere Virtual Enterprise)</li> </ul>	クライアント・リクエスト毎にDB2のワークロード管理を実施したい場合に検討して下さい。なお、AIXのCPU/プリフェッチの優先制御に紐付けることも可能です。
JPA (参考資料)	<ul style="list-style-type: none"> <li>•JPA</li> <li>• EJB 2.x Entity Bean</li> <li>•Hibernate</li> <li>•iBATIS</li> <li>•JDBC</li> </ul>	<p>以下を考慮し、どれを使用するかを決定して下さい。</p> <ul style="list-style-type: none"> <li>•O/Rマッピング技術を利用したいか</li> <li>•RuntimeはJPAをサポートしているか</li> <li>•標準に準拠しているか</li> <li>•実績を重視するか</li> <li>•EJB2.xの制約に耐えるか</li> </ul>

© 2012 IBM Corporation 70

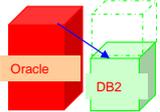
Designing Web System Infrastructure with WAS V8.0

## ■ □ IBM DB2 おすすめポイント !! □ ■

**ポイント①**

ライセンス・コストはOracleの半額以下

初年度保守、豊富な標準フィーチャー、開発用低価格ライセンス、スタンバイ・ライセンス不要

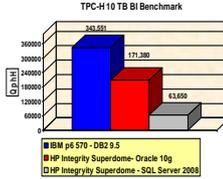


ライセンス料金

**ポイント②**

Power+DB2は最速のパフォーマンス

ベンチマーク結果でDB2は5年間以上首位をキープ！



TPC-H 10 TB BI Benchmark

■ IBM p6 570 - DB2 9.5  
■ HP Integrity Superdome - Oracle 10g  
■ HP Integrity Superdome - SQL Server 2008  
2009年7月7日現在

**ポイント③**

データ圧縮でストレージを大幅削減

他の追随を許さない圧縮率。ストレージ・コスト削減、バックアップ時間短縮。

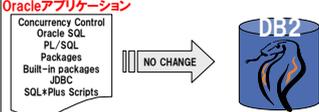


80%以上の圧縮率  
25TB → 5TB

**ポイント④**

既存Oracle資産とスキルの活用

これまで構築した旧バージョンのOraclePL/SQL資産、アプリケーション資産を保護。OracleMasterスキルがDB2でも利用可能



Oracleアプリケーション  
Concurrency Control  
Oracle SQL PL/SQL Packages  
Built-in packages  
JDBC  
SQL\*Plus Scripts

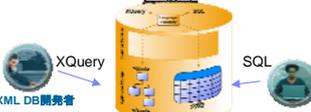
NO CHANGE

DB2

**ポイント⑤**

XMLとリレーショナルのハイブリッドDB

XMLは、アプリケーション変更が容易。銀行、証券、保険、医療など業界標準データ・フォーマット



XML DB開発者

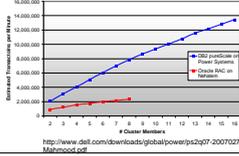
XQuery

SQL

**ポイント⑥**

拡張性と無停止を実現するpureScale

年間停止時間はOracleの10分の1以下。  
WLM、pureScaleとの組合せで更に堅牢に



pureScale vs. Oracle RAC Projected Transaction Scalability

Y-axis: Projected Transactions per Week (0 to 16,000,000)  
X-axis: # Client Machines (0 to 16)

Legend: pureScale (blue line), Oracle RAC (red line)

http://www.ibm.com/downloads/globai/powerps2q07-20070279-Mainmood.pdf

© 2012 IBM Corporation

71

## DB2 9.7 Technical Information

---

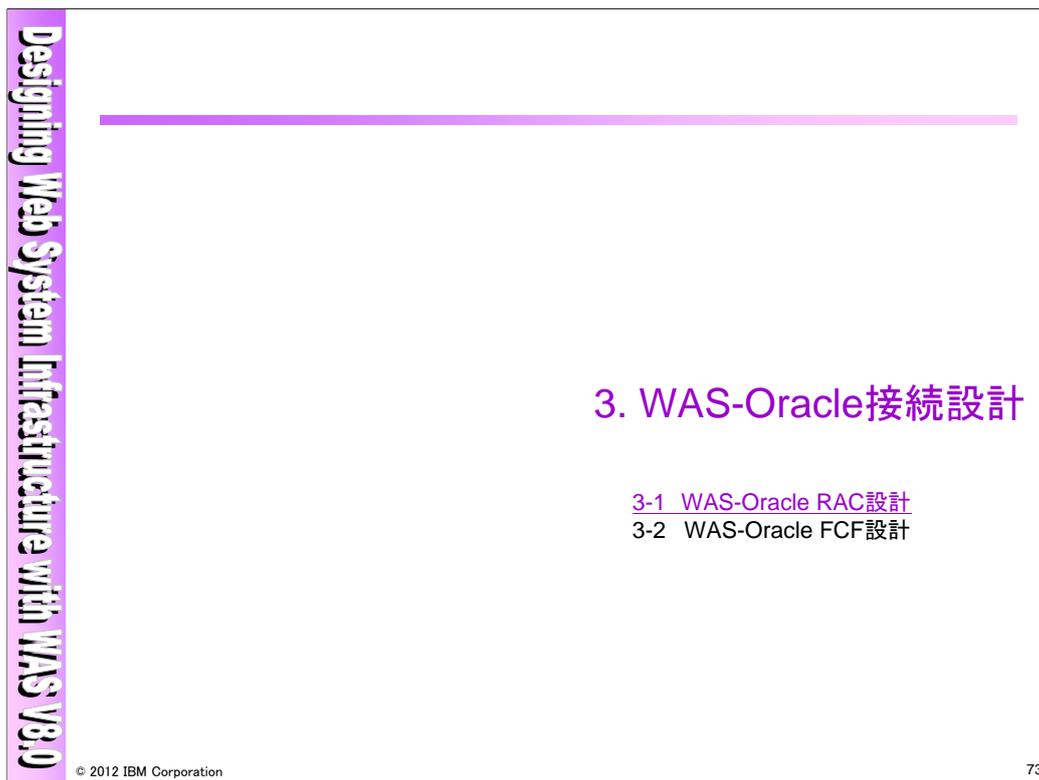
★★★ ご興味のある方はぜひこちらの資料をご覧ください！ ★★★

DB2Starイベント資料 「DB2 9.7 ー新機能の全貌ー」

[http://www.ibm.com/software/jp/data/event/db2star/pdf/B-1\\_noma.pdf](http://www.ibm.com/software/jp/data/event/db2star/pdf/B-1_noma.pdf)

DB2 V9.7 新機能ワークショップ

<http://www.ibm.com/developerworks/jp/data/products/db2/#DB2Z9.20>



3章では、WASからOracleにアクセスするための設計・設定についてご説明致します。はじめに、WAS-Oracle RAC設計についてご説明致します。

Designing Web System Infrastructure with WAS V8.0

## Oracle RAC (Real Application Cluster)概要

- 共有ディスク上に配置したひとつのデータベースを、複数のノードで稼動するOracleインスタンスから同時にアクセスできる「*Shared-Everything*」型のデータベース
- メリット
  - ◆ Active - Active 構成によるリソースの有効活用
  - ◆ 高速なフェールオーバーの実現
  - ◆ 高いスケーラビリティの実現
  - ◆ リクエストの負荷分散

The diagram illustrates the Oracle RAC architecture. At the top, two '共有ディスク' (Shared Disks) are shown. Below them, three 'Oracle' instances are depicted, each connected to the shared disks. A box labeled 'データベースとインスタンスの分離' (Separation of Database and Instance) points to the shared disks. Below the instances, two 'クライアント' (Clients) are shown, connected to the Oracle instances. A box labeled 'Oracle' is also present. Callouts provide additional details: 'Active x Active構成 1つの統合されたサーバーとして、仮想化する' (Active x Active configuration, virtualized as one integrated server) and 'リスナー機能 クライアントからの接続要求に応答し、RACインスタンスとの接続を確立する' (Listener function: responds to client connection requests and establishes connection with RAC instances).

74

Oracle RACは、Real Application Clusterの略で、共有ディスク上に配置したひとつのデータベースを、複数のノードで稼動するOracleインスタンスから同時にアクセスできる「Shared-Everything」型のデータベースです。主に以下の利点があります。

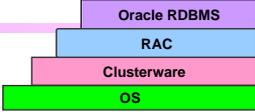
- リソースの有効活用  
Active/Active構成が可能です。
- 高速なフェールオーバーの実現  
ダウンを自動で検知し、Activeノードへフェールオーバーします。
- 高いスケーラビリティの実現  
アクセスが集中する時期だけ、リソースを割り当てることが可能です。  
接続先リストにリソースを追加することが可能です。
- リクエストの負荷分散  
リソースの使用状況に合わせてワークロードを管理できます。

また、Oracle RACは、Oracleデータベース(データを格納する領域)とOracleインスタンスが分離されています。そして、Oracleデータベースを共有ディスクとして定義し、Oracleインスタンスをサービスに関連付け仮想化します。仮想化されたOracleインスタンスは、1つの統合されたデータベースサーバーとして参照することができます。

Designing Web System Infrastructure with WAS V8.0

## WAS - Oracle RAC 接続設計 (1/3)

- サポート・バージョン (WASV8)
  - ◆ WAS側 Oracle 10gR2 / Oracle 11gR1,R2
  - ◆ Oracle側 Oracle 10g / Oracle 11gにて、RAC構成が実装



- JDBCドライバー選択指針とデータ・ソース設定
  - ◆ JDBC Thinドライバー
    - ▶ Type4接続: Oracleネットワーク・プロトコルのPure Java実装経由でDBに接続
    - ▶ Oracleクライアントが不要、設定がシンプル
    - ▶ データ・ソースのURL設定 ⇒ `jdbc:oracle:thin:@(HOST_NAME):(PORT_NUMBER):(SID)`  
 \* 接続先(listener.oraに定義したホスト名とポート番号)を指定
  - ◆ JDBC OCI ドライバー
    - ▶ Type2接続: Oracle Call Interface経由でデータベースに接続する
    - ▶ Oracleクライアントが必要、機能がThinドライバーと比べて豊富
    - ▶ SolarisおよびWindows用のみ提供
    - ▶ データ・ソースのURL指定 ⇒ `jdbc:oracle:oci:@(SERVICE_NAME)`  
 \* 接続先(tnsnames.oraに定義したサービス名)を指定

**DB2の場合**

共通データ・ソース・プロパティ および必須データ・ソース・プロパティ

名前	値
ポート番号	50000
サーバー名	192.168.0.1
ドライバー・タイプ	4
データベース名	sample

**Oracleの場合**

共通データ・ソース・プロパティ および必須データ・ソース・プロパティ

名前	値
URL	jdbc:oracle:thin:@(DESCRIPTIC

Oracle RAC機能はOracle 9i Enterprise、Oracle 10g Standard/Enterprise、Oracle 11g standard/Enterpriseにて提供されています。WASV8は、Oracle 10gR2とOracle 11gR1/R2をサポートします。以下のSystem Requirementsをご確認下さい。

• System Requirements for WebSphere Application Server V8.0

[http://www.ibm.com/support/docview.wss?uid=swg27021246#AIX\\_Databases\\_wv](http://www.ibm.com/support/docview.wss?uid=swg27021246#AIX_Databases_wv)

Oracle社が提供するクラスター・ソフトウェアは、Oracle 9iでは有償でしたが、Oracle 10g R1より無償となりました。これを使用することで、障害対策のための高度なアーキテクチャを実現することが可能になります。また、サードベンダーが提供するクラスター・ソフトウェアを使用することも可能です。ただし、Oracle 10g Standard EditionのRAC (SERAC)を使用される場合には、Oracle社が提供するクラスター・ソフトウェア (Oracle Clusterware)を使用する必要があります。

JDBCドライバーとして、ThinドライバーとOCIドライバーがあります。ThinドライバーはJDBCタイプ4、OCIドライバーはJDBCタイプ2のドライバーです。OCIドライバーは、Thinドライバーより機能が豊富ですので、機能を重視する場合にご選択下さい。しかし、Oracleクライアントをインストールするといった作業が発生しますので、問題判別の際には、ケアするポイントが多くなる構成ともいえます。

• Oracle JDBC Driver 11.2 ダウンロードサイト

<http://www.oracle.com/technetwork/jp/database/enterprise-edition/jdbc-112010-278104-ja.html>

Oracle社が提供するOCIおよびThinドライバーの詳細は、以下を参照して下さい。

• Oracle Database JDBC開発者ガイド 11gリリース2 (11.2)

[http://docs.oracle.com/cd/E16338\\_01/java.112/b56281/overvw.htm](http://docs.oracle.com/cd/E16338_01/java.112/b56281/overvw.htm)

**Designing Web System Infrastructure with WAS V8.0**

### WAS JDBC Provider Type と DataStoreHelper クラスの対応関係 =Oracleの場合=

- WAS V8 は JDK 6.0上で稼動するため、  
Oracle JDBC Driver としては ojdbc6.jar ファイルを利用する
  - ◆ Oracle 11gのJDBCドライバーは、JDBC4.0標準がサポートされる
  - ◆ Oracle 10gのJDBCドライバーは、JDBC3.0標準がサポートされる
- WAS V8 で ojdbc6.jar を利用する場合には、OracleDBサーバーが10gでも11gでも、Oracle11gDataStoreHelperを使う必要がある
  - ◆ WAS V8 ではOracle10gDataStoreHelperクラス、OracleDataStoreHelperクラスはDeprecated扱いとなっている
- WAS V7 の場合も、Oracleバージョンにかかわらず(10g,11g) ojdbc6.jar ならびに Oracle11gDataStoreHelperクラスを使用する

© 2012 IBM Corporation 76

WAS V8がサポートするOracle Databaseは10gと11gです。

Oracle JDBC Driver は、10g、11gそれぞれでJDK5対応のojdbc5.jarとJDK6.0対応のojdbc6.jar とが提供されていますが、

WAS V8はJDK6.0上で動作していますので ojdbc6.jar を使用してください。

また、同じ ojdbc6.jar でも、どのバージョンのOracle JDBC Driverを使うかでJDBCバージョンへの対応状況が異なります。

Oracle10gのojdbc6.jarではJDBC3.0を、Oracle11gのojdbc6.jarではJDBC4.0標準をサポートしています。

データストアヘルパークラスは、Oracleバージョンによらず(10g/11g)、Oracle11gDataStoreHelperクラスが必須となっています。

・WAS V8.0 InfoCenter「Oracle 用データ・ソースの最小必要設定」

[http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=%2Fcom.ibm.websphere.nd.doc%2Finfo%2Fae%2Fae%2Frdat\\_minreqoracle.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=%2Fcom.ibm.websphere.nd.doc%2Finfo%2Fae%2Fae%2Frdat_minreqoracle.html)

Designing Web System Infrastructure with WAS V8.0

## WAS - Oracle RAC 接続設計 (2/3)

- データ・ソース設定例
  - ◆ THINドライバ利用時のURL指定 ⇒

共通データ・ソース・プロパティおよび必須データ・ソース・プロパティ	
名前	値
* URL	jdbc:oracle:thin:@(DESCRIPTION=

```

jdbc:oracle:thin:@(DESCRIPTION=
  (ENABLE=BROKEN)
  (ADDRESS_LIST=
    (FAILOVER=ON)
    (LOAD_BALANCE=ON)
    (ADDRESS = (PROTOCOL = TCP)(HOST = rac1vip)(PORT = 1521))
    (ADDRESS = (PROTOCOL = TCP)(HOST = rac2vip)(PORT = 1521))
  )
  (CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=rac))

```

→ 接続失敗時に別のサーバーに試行  
 → 接続先への負荷分散  
 } → 接続先のリスト  
 → サービス定義

```

Listener.ora
LISTENER_RAC1 =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = rac1vip)(PORT = 1521)(IP = FIRST))
      (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.0.1)(PORT = 1521)(IP = FIRST))
    )
  )

```

77

上記スライド部分を参考にして、データ・ソースのプロパティを設定して下さい。

「URL」に、Oracleのサービスを定義します。また、LOADBALANCE=ONの場合は、接続先リストにあるDBにランダムに割り振られます。LOADBALANCE=OFFの場合は、接続先リストに指定した順に割り振られます。

Designing Web System Infrastructure with WAS V8.0

## WAS - Oracle RAC 接続設計 (3/3)

- **Single Client Access Name (SCAN: 11gR2 RAC 新機能)**
  - ◆ Oracle RAC クラスタ内のデータベースへ接続する際の単一のエイリアス
  - ◆ クライアントからSCAN名を指定するだけでOracle RAC 内いずれかのインスタンスへと接続できる
  - ◆ SCANリスナーへの接続確立後は、SCANが各サービスへの接続を自動的にリダイレクトする。
  - ◆ 11gR1以前のOracle RACでは、VIPをサーバー台数分指定することで負荷分散やフェールオーバーを実現
    - ⇒ 11gR2 RACでは、負荷分散・フェールオーバーの機能がSCANにより提供される。(DBクライアント側では、RACノード台数・VIP名等を意識する必要がない。)

© 2012 IBM Corporation 78

従来のOracle RAC を使うシステムでは、RACノードの数だけWASデータソースのURL定義にOracleリスナーのListenするホスト名、ポートのペアを定義する必要があります。

Oracle RAC 11gR2で登場した新機能のSCANリスナーが使われるシステムでは、WASデータソースのURL定義に記述するホスト名、ポートのペアは1組でよく、LoadBalance, FailoverはOracle RAC SCANリスナー側で制御されます。

SCANを利用するOracle RACのシステムに接続する際には、WAS側にはSCANのホスト名、ならびにポート番号を指定しておきます。

SCANのホスト名は、Oracle側で複数のIPアドレスに対応づけられており(推奨は3つ)、DNSラウンドロビンによって負荷分散が実現される仕組みです。

SCANが利用されるシステムでは、初期構築時やノード追加・削除などのタイミングでDBクライアントに対して全ての接続先サーバーへの接続定義を登録する必要がないという運用上の恩恵が得られるということです。

このような仕組みを使っているかどうかはOracle担当者が把握している内容となりますので、Oracle担当者と密に連携し、設計やテスト計画立案等の作業を進めるようにしてください。

Designing Web System Infrastructure with WAS V8.0

## WAS - Oracle RAC 設計 考慮点 (1/2)

- 失効した接続オブジェクトの課題
  - ◆ 接続プール内には複数データベースへの物理接続が共存する
  - ◆ 片系のデータベースがダウンした場合、ページ・ポリシーをEntirePoolに設定していると、有効な接続オブジェクトも破棄される
- 設定指針
  - ◆ ページ・ポリシー: FailingConnectionOnly
  - ◆ Oracle FCF(→後述)

79

Oracle RACを構成した環境では、接続プール内には複数のデータベースとの物理接続が共存した状態になります。この状態で、一方のデータベースがダウンした場合、ダウンしたデータベースに紐付いている接続オブジェクトのみが失効しますが、ダウンしていないデータベースと紐付く接続オブジェクトは有効です。しかし、クライアントが失効した接続オブジェクト取得し、StaleConnectionExceptionが発生した場合、接続プール内の全ての接続オブジェクトがページ対象となります。従いまして、Purge PolicyをEntirePoolに設定していると、接続先データベースの状態に関わらず、接続プール内の全ての接続オブジェクトが破棄されてしまいます。

有効な接続オブジェクトを破棄したくない場合は、Purge PolicyをFailingConnectionOnlyに設定することをご検討下さい。また、Oracle FCFの機能を使用することで、当問題を回避できますので、後述致します。

Designing Web System Infrastructure with WAS V8.0

## WAS - Oracle RAC 設計 考慮点 (2/2)

- 2フェーズ・コミット処理は、異なるRACインスタンス上で処理が実行される可能性があり、Oracle RACの制限に抵触する

**Oracle RAC#1**  
1.xa\_start  
2.SQL処理の実行  
3.xa\_end  
4.xa\_prepare  
ここで障害が発生

**Oracle RAC#2**  
1.xa\_commit  
異なるインスタンス上で、xa\_commitが実行される

**設定指針**

- ◆ Oracle 11gの場合 GLOBAL\_TXN\_PROCESSES(デフォルト1)の設定
- ◆ Oracle 10gの場合 DTPサービスの構成

DTPサービスの構成については、巻末資料「DB接続設計-参考-」をご参照下さい。

80

Oracle RACを構成した環境において、2フェーズ・コミットを実行すると、異なるRACインスタンスで処理が実行される可能性があります。

ノード間で同一ブランチの xa\_prepare/xa\_rollback/xa\_commit 要求を実行することができないという制限があり、実際に異なるRACインスタンスで処理が実行されてしまった場合は、トランザクションはそのインスタンス上では存在していないため、エラーが発生します。

この問題の対応策として、Oracle10gの環境ではDTPサービスを構成する必要があります。

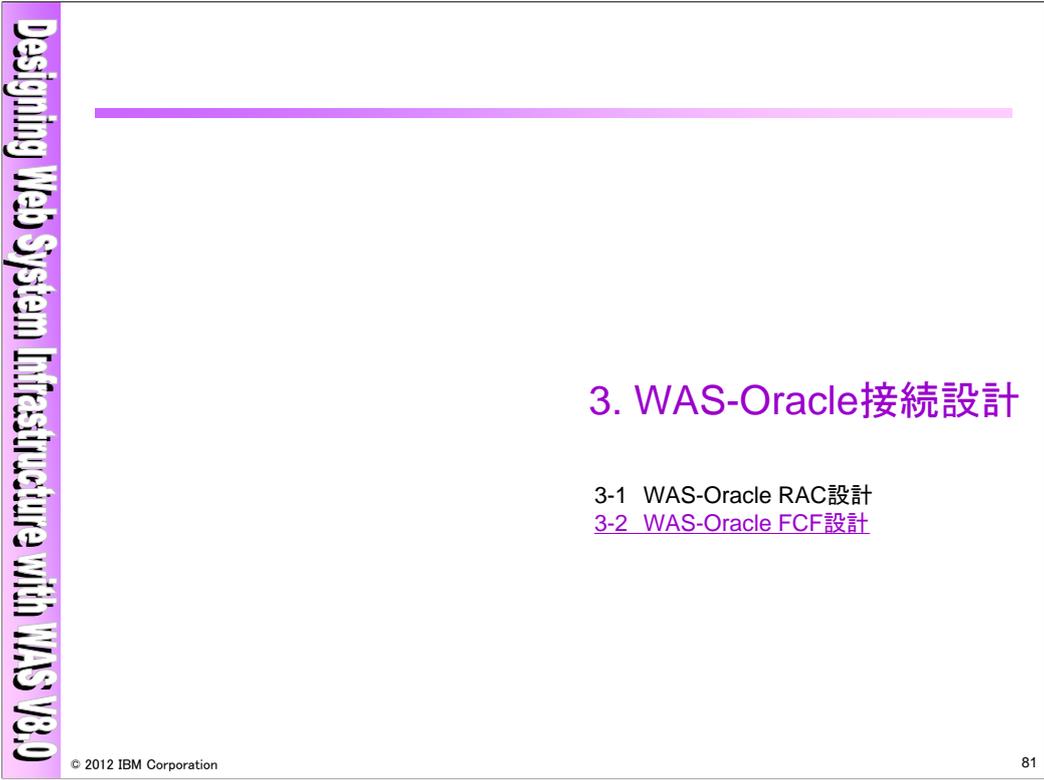
Oracle11gの環境では、GLOBAL\_TXN\_PROCESSESがデフォルトで設定されているため、特に考慮する必要はありません。

エラー内容:

ORA-24756: トランザクションが存在しません。

原因: 無効なトランザクション識別子またはコンテキストが使用されました。または、トランザクションは完了しています。

処置: トランザクションが完了していない場合は、有効な識別子を指定してコールを再実行して下さい。



Designing Web System Infrastructure with WAS V8.0

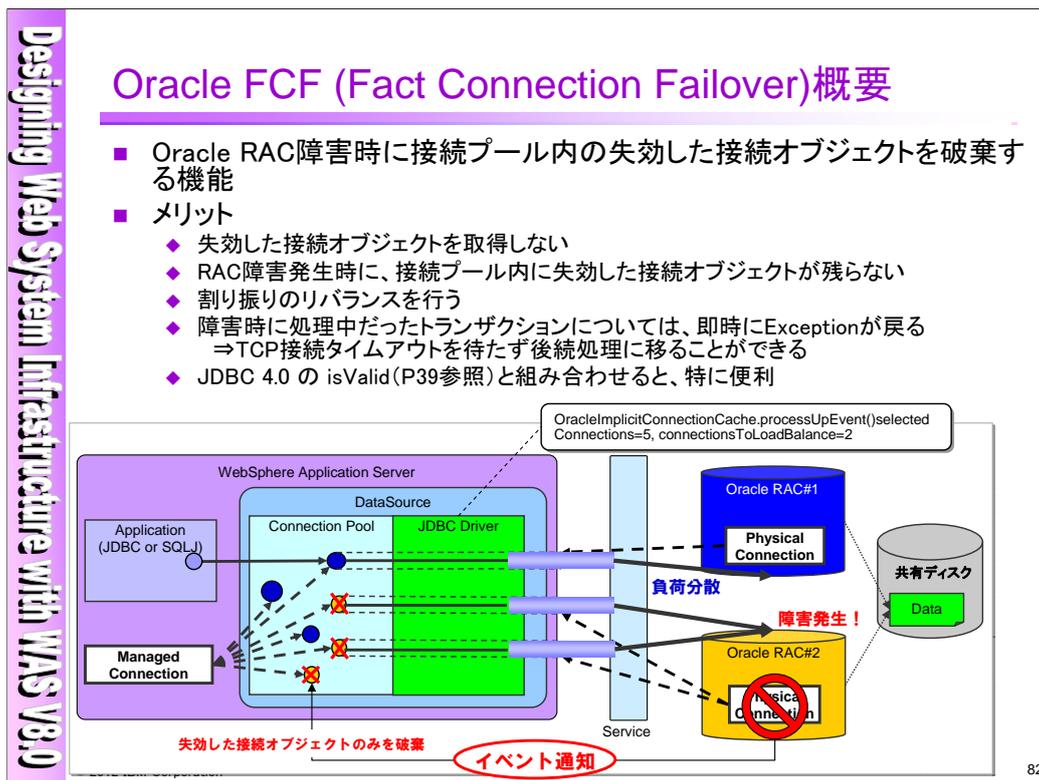
---

### 3. WAS-Oracle接続設計

- 3-1 WAS-Oracle RAC設計
- 3-2 WAS-Oracle FCF設計

© 2012 IBM Corporation 81

続きまして、WAS-Oracle FCF接続設計についてご説明致します。



FCFとはFast Connection Failover の略で、あるRACインスタンスに障害が発生した場合に、クライアントとなるWASに障害が発生したことを通知し、接続プール内の失効した接続オブジェクトを破棄することで、高速にフェイルオーバーする技術です。この機能を使用することで、接続プール内から無効なコネクションを取得することがなくなります。また、割り振りのリバランスの機能も備えられています。片方のRACインスタンスに障害が発生し回復すると、接続プールに蓄積されている接続オブジェクトは接続しなおさないで、割り振りの偏りが発生します。FCFはこの割り振りの偏りを検知し、自動でこの偏りを修正することができます。

Oracle RAC構成の場合、この割り振りの偏りが発生する可能性がありますので、即座に偏りを解消したい場合には、以下の対応策を実施して下さい。

- Oracle FCFを使用する
- 経過タイムアウトを設定し、定期的に接続オブジェクトを破棄する
- WASを再起動する

Designing Web System Infrastructure with WAS V8.0

## WAS - Oracle FCF接続設計

- サポート・バージョン
  - ◆ WAS側 Oracle 10g / Oracle 11g
    - WASのデータ・ソースを無効化する
  - ◆ Oracle側 Oracle 10g R1以降
    - RAC環境/OracleRestart環境(⇒次ページ)にて利用可能
- JDBCドライバー設計
  - ◆ 実装クラス名に、*oracle.jdbc.pool.OracleDataSource*を設定する
    - 2フェーズ・コミット処理(*javax.sql.XADataSource*)には対応していません
- データ・ソース設計
  - ◆ UCP(Universal Connection Pool、旧実装:ICC)の有効化
    - Connection Cache内の管理を行う
      - データ・ソースの作成、キャッシュの生成/破棄
        - ✓ `connectionCachingEnabled => true`
  - ◆ FCFの有効化
    - `FastConnectionFailoverEnabled => true`
- ONS(Oracle Notification Service)の構成
  - ◆ 障害発生/復旧に関するイベントの伝播を行う
    - `{ORACLE_HOME}/opmn/conf/ons.config`

© 2012 IBM Corporation 83

Oracle FCF構成の前提条件は、下記Oracleホームページをご確認下さい。

- ・ Oracle Database JDBC開発者ガイドおよびリファレンス - 「高速接続フェイルオーバー」

[http://otndnld.oracle.co.jp/document/products/oracle10g/102/doc\\_cd/java.102/B19275-03/fstconfo.htm](http://otndnld.oracle.co.jp/document/products/oracle10g/102/doc_cd/java.102/B19275-03/fstconfo.htm)

[http://docs.oracle.com/cd/E16338\\_01/java.112/b56281/fstconfo.htm](http://docs.oracle.com/cd/E16338_01/java.112/b56281/fstconfo.htm)

FCFを利用する場合は、WASの接続プール機能は無効化し、Oracleが提供するConnection Cache機能を利用します。従いまして、WASが提供する接続プールの機能は使用できませんので、TPVを使用できなくなります。また、FCFの制限として、2フェーズ・コミット処理は実行できません。

Oracle FCFを使用するにあたり、Oracle JDBC Driverに既知の問題(バグ #6638862)があります。詳しくは、Oracle社にお問い合わせ下さい。

- ・ WASV8.0 Information Center - 「アプリケーション・サーバーでの Oracle 接続キャッシュの構成」

[http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tat\\_oracleracconnpool.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tat_oracleracconnpool.html)

また、この問題は、TechNoteにてコーディングによる回避策が紹介されています。

- ・ TechNote - 「The Oracle database does not cache all of the connections」

<http://www.ibm.com/support/docview.wss?rs=180&uid=swg21318192>

Designing Web System Infrastructure with WAS V8.0

## (参考) OracleRestart

- OracleRestart とは
  - ◆ シングル・インスタンス環境の高可用性を実現する構成
  - ◆ Oracle Clusterware上でOracleを起動し、プロセス監視を行う
  - ◆ プロセス障害時にはOracle自身の仕組みでプロセス再起動が行われる
  - ◆ FCFもサポートされる
  - ◆ Oracle 11gR2 シングル・インスタンス構成 = OracleRestart となるわけではない
    - ▶ OracleASMを利用する場合には前提SWとしてOracle Clusterwareが導入され、OracleRestart構成となる
    - ▶ Oracle Grid Infrastructure (Clusterware) を明示的に導入せず、ファイル・システムにてOracleDBを作成した場合にはOracleRestartの無い構成となる

スタンド・アローン構成

スタンド・アローン
DBサーバー1台構成
DBサーバーのホスト名 (IP)、Portは常に同一
DBサーバーの復旧までサービスが完全停止

© 2012 IBM Corporation 84

もうひとつの新しい構成として、スタンド・アローンであるシングル・インスタンスに高可用性機能が付加された構成として、OracleRestart が提供されるようになりました。Oracle Clusterwareによるプロセス監視が行われ、プロセス障害時にはプロセス再起動が試行されます。

Oracle Clusterwareが前提として導入される、OracleASMを利用する際にはOracleRestart構成となりますが、ファイル・システム・ベースでOracleを構築する場合にはこの構成となりません。(OracleRestartが必要であればそのように構成することもできます)

(※) OracleASM:

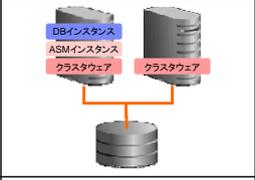
Oracle独自のボリューム・マネージャー機能。ASMインスタンスにより、ストライピングやミラーリングなどのボリューム管理が実現される。

Designing Web System Infrastructure with WAS V8.0

## (参考) Oracle RAC One Node

- Oracle RAC One Nodeとは
  - ◆ OracleRACで使われてきたクラスターウェア (Oracle Clusterware)による Oracle DB コールド・スタンバイ構成
    - ▶ 従来のOracleDBコールドスタンバイは 3rd Party HAソフトウェアにて実現
    - ▶ これからも3rd Party HA ソフトウェアによる実装は引き続き可能
  - ◆ ”Oracle RAC”とは異なり、一時点でア クティブとなるOracleDBノードは1台のみ

**コールド・スタンバイ構成 (HA構成)**



アクティブ-スタンバイ型
一時点で稼動するDBサーバーは1台となる構成。サーバー機器障害に備えた冗長化構成
DBサーバーのホスト名 (IP)、Portは常に同一になる構成とする
IP/ディスクの引き継ぎが完了し、スタンバイ側のDBが起動するまでサービス停止となる

© 2012 IBM Corporation 85

RAC One Nodeは、Oracle Clusterwareにより実装されるコールドスタンバイ構成のことです。

1ノードで実装されるOracle RACとも言え、FCFなどOracleRACで使える機能は基本的に使うことができます。

「RAC」とつくのでアクティブ-アクティブ構成のOracle RACと少し紛らわしいですが、その名の通りデータベースが稼動するノードは1時点で1台のみですので、WASデータ・ソースを構成する上で意識する必要がある接続先DBホスト、ポート番号は基本的にひとつとなります。

Designing Web System Infrastructure with WAS V8.0

### まとめ: WAS-Oracle接続設計

項目	選択肢	設計指針
JDBCドライバー	<ul style="list-style-type: none"> <li>•JDBC Thinドライバー</li> <li>•JDBC OCIドライバー</li> </ul>	基本的に、データベース・ベンダーの指示に従って下さい。
失効した接続オブジェクト	<p>&lt;製品機能/実装による自動対応&gt;</p> <ul style="list-style-type: none"> <li>•Oracle FCF</li> <li>•接続検証プロパティ</li> <li>•アプリケーション実装</li> </ul> <p>&lt;運用による手動対応&gt;</p> <ul style="list-style-type: none"> <li>•WAS再起動</li> <li>•管理コンソール</li> <li>•wsadminコマンド</li> </ul>	<p>Oracle RAC構成にて、失効した接続オブジェクトのみをパージしたい場合には、FailingConnectionOnlyの設定かOracle FCFを使用することを検討して下さい。</p> <p>失効した接続オブジェクトの破棄方法は、以下より検討して下さい。</p> <ul style="list-style-type: none"> <li>•JDBC4.0環境 JDBCドライバーによる検証</li> <li>•JDBC3.0環境 StaleConnectionExceptionをcatch (SQL照会による検証)</li> <li>•Oracle FCF</li> </ul>
2フェーズ・コミット処理 (Oracle RAC)	<ul style="list-style-type: none"> <li>•DTPサービス</li> <li>•GLOBAL_TXN_PROCESSES</li> </ul>	Oracle 10g環境ではDTPサービスを構成し、Oracle 11g環境ではGLOBAL_TXN_PROCESSESを設定する。GLOBAL_TXN_PROCESSESは、デフォルトで設定されています。
失効した接続オブジェクト / 割り振りリバランス	<ul style="list-style-type: none"> <li>•Oracle FCF</li> </ul>	Oracle FCFにて、失効した接続オブジェクトのみの破棄が可能です。また、RAC構成における割り振りのリバランス機能も備えられています。しかし、WASのデータ・ソースが使用できない、2フェーズ・コミット処理ができない、バグが公開されている等の考慮点があります。

© 2012 IBM Corporation 86

**Designing Web System Infrastructure with WAS V8.0**

---

まとめ・参考文献

© 2012 IBM Corporation

87

**Designing Web System Infrastructure with WAS V8.0**

### まとめ

- WAS V8.0とデータベースを使用したシステムのコンポーネントとトポロジー構成
- WAS V8.0とDB2との接続における設計手法
  - ◆ JDBCドライバー設計指針
    - DB2のJDBCドライバー、DataStoreHelper
  - ◆ データソース設計指針
    - セキュリティー(認証情報の指定方法、Trusted Context)、接続プール、データソース・プロパティ(キャッシュ、失効した接続オブジェクトの対応、DB2 Client Reroute、拡張DB2データソース)
  - ◆ アプリケーション設計指針の提示
    - アノテーション、共用スコープ、分離レベル
  - ◆ パフォーマンス / 問題判別
    - 実行時間の測定方法、WASTレース設定
  - ◆ Hints & Tips
    - clientInformationAPI、DB2 64bitインスタンスへの接続方法、JPA
- WAS V8.0とOracleとの接続における設計手法
  - ◆ WAS - Oracle RAC接続設計
  - ◆ WAS - Oracle FCF接続設計

© 2012 IBM Corporation 88

当セッションでは、WAS V8.0とデータベースを使用したシステムのコンポーネントとトポロジー構成について、WAS V8.0とDB2との接続における設計手法、WAS V8.0とOracleとの接続における設計手法についてご説明致しました。

WAS V8.0にてシステム品質を向上させるDB接続に関する様々な機能が追加されております。お客様要件に合う場合には、ご紹介した新機能を採用されることをご検討下さい。

## 参考文献

- Information Center
  - ◆ WebSphere Application Server V8.0
    - <http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp>
  - ◆ IBM DB2 Database for Linux, UNIX, and Windows V9.7
    - <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp>
- ワークショップ資料
  - ◆ WebSphere Application Server V8 アナウンスメント・ワークショップ
    - [http://www.ibm.com/developerworks/jp/websphere/library/was/was8\\_ws/](http://www.ibm.com/developerworks/jp/websphere/library/was/was8_ws/)
  - ◆ WebSphere Application Server V7による基幹システム設計ワークショップ
    - [http://www.ibm.com/developerworks/jp/websphere/library/was/was7\\_guide/index.html](http://www.ibm.com/developerworks/jp/websphere/library/was/was7_guide/index.html)
  - ◆ DB2 9.7 技術資料
    - <http://www.ibm.com/jp/domino01/mkt/dminfo.nsf/doc/001BED48>
    - <http://www.ibm.com/jp/domino01/mkt/dminfo.nsf/doc/001C70DF>
    - <http://www.ibm.com/jp/domino01/mkt/dminfo.nsf/doc/001DAA42>
    - <http://www.ibm.com/jp/domino01/mkt/dminfo.nsf/doc/001DDA9D>
    - <http://www.ibm.com/jp/domino01/mkt/dminfo.nsf/doc/001B138C>

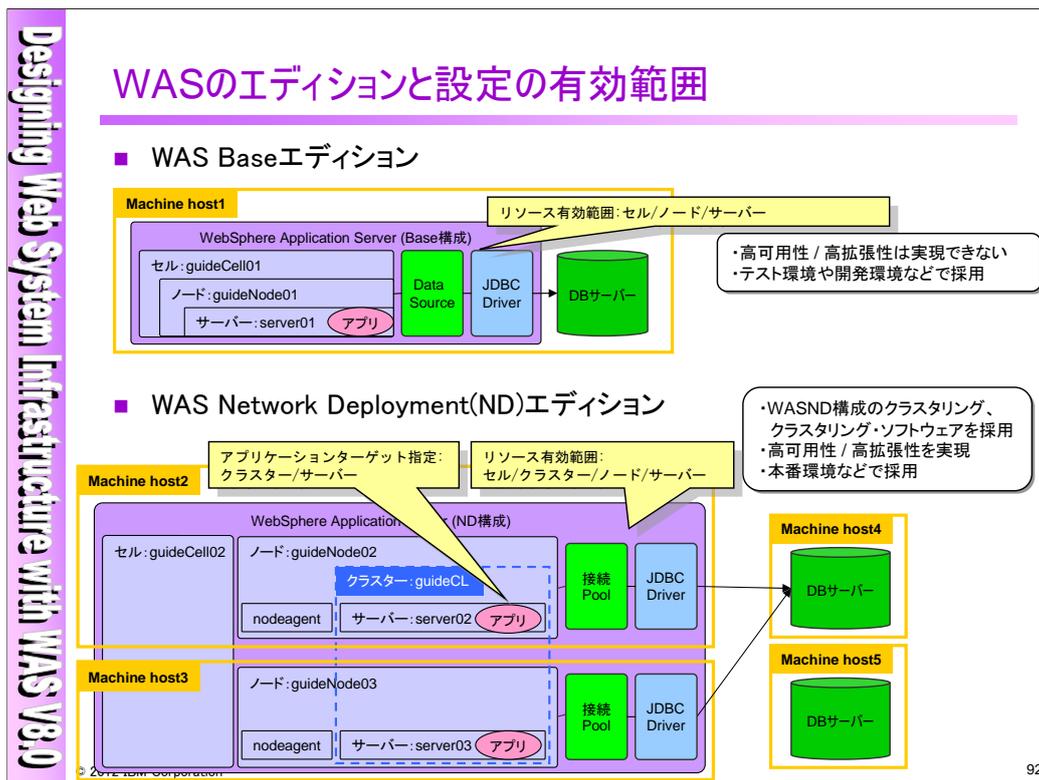
Designing Web System Infrastructure with WAS V8.0

WASV8.0によるWebシステム基盤設計Workshop

DB接続設計  
－ 参考 －



1章では、WebSphere Application Server(以降、WAS)からデータベースにアクセスする際に必要となるコンポーネントと、WASとデータベースを使用したシステムのトポロジーについてご説明致します。



WASからデータベースにアクセスするためには、JDBCドライバー、データ・ソース、アプリケーションに必要な設定を行います。詳細な設定は、本資料の2章(WAS-DB2接続設計)にてご説明しておりますが、JDBCドライバー、データソース、アプリケーションに設定した内容がどの範囲で有効となるかを考慮する必要があります。JDBCドライバーとデータ・ソースにおいては、セル / クラスタ / ノード / サーバーの4つの有効範囲を選択することができますが、WASV6以降はセル・レベルでのリソース定義は非推奨ですので、ご注意ください。詳細は下記リンク先をご確認下さい。また、有効範囲を選択するに当たっては、以下をご留意下さい。

- ・セキュリティ面を考慮すると、そのリソースを使用しないメンバーから参照出来ないように極力狭い範囲の範囲で設定する事が望ましい
- ・運用管理面を考慮すると、極端に狭すぎる範囲でのリソース定義は、定義に変更があった場合に修正箇所が増え運用負荷が高くなる
- ・WASV8.0 Information Center - 「非推奨のフィーチャー、安定化されたフィーチャー、および除去されたフィーチャー」

[http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/rmig\\_deprecationlist.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/rmig_deprecationlist.html)

- WAS設定の有効範囲
- JDBCドライバーの設定
- WAS V7/V8がサポートするJDBCドライバー
- DB2用のDataStoreHelperクラス

## 2. WAS-DB2接続設計

### ・WAS-DB2接続設計

- JDBCドライバー設計
  - データソース設計
  - アプリケーション設計
  - パフォーマンス / 問題判別
  - Hints & Tips
- ・WAS-Oracle接続設計

Designing Web System Infrastructure with WAS V8.0

## JDBCドライバーの設定

### ■ 管理コンソール

JDBCドライバー作成時に選択した値が、そのまま反映される。  
クラス・パス、ネイティブ・ライブラリー・パスに注意する。

### ■ JDBCドライバーのバージョン確認

- ◆ SystemOut.log
- ◆ Javaコマンド

JDBCドライバー名とバージョン情報

```

[省略] DSRA8203: Database 製品名 : DB2/AIX64
[省略] DSRA8204: Database 製品バージョン : SQL09050
[省略] DSRA8205: JDBC driver 名 : IBM Data Server Driver for JDBC and SQLJ
[省略] DSRA8206: JDBC driver バージョン : 4.1.85
[省略] DSRA8212: DataStoreHelper 名 :
com.ibm.websphere.rsadapter.DB2UniversalDataStoreHelper@11a411a4。
                    
```

DataStoreHelper名

```

# java com.ibm.db2.jcc.DB2Jcc -version (db2jcc.jarの場合)
IBM DB2 JDBC Universal Driver Architecture 3.50.143
# java com.ibm.db2.jcc.DB2Jcc -version (db2jcc4.jarの場合)
IBM Data Server Driver for JDBC and SQLJ 4.0.90
                    
```

-configurationオプションを指定すると、さらに詳細な情報を確認できる

© 2012 IBM Corporation

94

JDBCドライバーは、以下のリンク先などをご参考に、管理コンソールから有効範囲、JDBCプロバイダー名、クラス・パス、ネイティブ・パス、実装クラス名などを設定し、新規作成して下さい。設定後の管理コンソールの画面が上記スライド部分になります。

・WASV8.0 Information Center - 「DB2 用データ・ソースの最小必要設定」

[http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/topic/com.ibm.websphere.nd.multipatform.doc/info/ae/ae/rdat\\_minreqdb2dist.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/topic/com.ibm.websphere.nd.multipatform.doc/info/ae/ae/rdat_minreqdb2dist.html)

JDBCドライバーのバージョンは、実際にWASからDB2に接続した際(WAS起動後の初回接続時)に、SystemOut.logに出力されます。また、DBサーバー上でJavaコマンドを実行することでも確認できます。

Designing Web System Infrastructure with WAS V8.0

## WAS V7/V8がサポートするJDBCドライバー

■ WAS V7/V8 のJDBCドライバー・サポート状況は下表の通りです。

WAS: JDBC Provider Type	JDBC Version	DB2 Version	DB2 8.2以前 *サポート終了	DB2 9.1	DB2 9.5	DB2 9.7
DB2 Legacy CLI-based Type 2 JDBC Drivers (サポート終了)	2.0	DB2 Legacy CLI-based Type 2 JDBC Drivers (db2java.zip)	サポート終了	非推奨 (WAS V7/V8は非サポート)		
DB2 Universal JDBC Driver Provider	2.0	DB2 Universal JDBC driver (db2jcc.jar)	サポート終了			
	3.0	DB2 Driver for JDBC and SQLJ (db2jcc.jar)		サポート有		
DB2 Using IBM JCC Driver	4.0	DB2 Data Server Driver for JDBC and SQLJ (db2jcc.jar)				
		DB2 Data Server Driver for JDBC and SQLJ (db2jcc4.jar)				サポート有

※ Universal / DB2 Driver / JCC / Data Server Driver は名前が異なるのみで、実体としては同じです。

95

WAS V7がサポートするJDBCドライバーは深緑で、WAS V8がサポートするドライバーについては紺色で示しています。

この一覧表においてWAS V7 がサポートするJDBCドライバーの種類が多いように見えるのは、DB2のバージョンアップに伴い、JDBCドライバの名前が変わっているためです。2012年2月現在WAS V7は、DB2 V9.1/V9.5/V9.7 の提供するJDBCドライバーをサポートしています。

Designing Web System Infrastructure with WAS V8.0

## DataStoreHelper

- SQLCodeとSQLStateから、例外の種類を判別するためのマッピング情報を持っているクラス
  - DB2に接続する場合のクラス名は「DB2UniversalDataStoreHelper」

- 選択指針
  - JDBCドライバー毎に提供されているDataStoreHelperを選択して下さい。
- エラー検出モデル
  - 例外マッピング・モデルの使用
    - デフォルト値、DataStoreHelperのエラーマップに定義された例外で置換する
  - 例外検査モデルの使用
    - DataStoreHelperのエラーマップに定義された例外で置換しない

java.sql.SQLRecoverableException: DSRA9110E: Connection はクローズされています。

© 2012 IBM Corporation 96

DataStoreHelperクラスは、初回のコネクション確立時と使い終わったコネクションをプールへ戻す際に必要な処理や、エラーが発生した際にDBから戻される、ベンダー固有のエラーコードをWebSphere特有のエラークラスへのWrapping等、各RDBMS固有となる処理を行うためのプラグインの役割を果たします。

DataStoreHelperは、SQLCodeとSQLStateから、例外の種類を判別するためのエラー・マッピング情報を持っています。このマッピング情報が必要なのは、同じ問題を意味しているエラーがデータベース・ベンダーによって、異なるSQLエラーおよびコードで提供されていることがあるためです。

図示したように、例えば接続の障害を表すコードはDB2とOracleとで、数も、エラーコードを示す数字そのものもまったく異なりますが、DataStoreHelperによって、WebSphereランタイムにはStaleConnectionExceptionというエラーとなり、プログラム・コードで状況に応じたハンドリングが出来るようになっています。DBベンダ固有のオリジナルのエラーコードはアプリ側でgetErrorCodeメソッドを実行することで確認することができます。

マッピング情報の詳細は、各DataStoreHelperクラスのJavadocにてご確認下さい。

WAS V8.0 InformationCenter - 「Class GenericDataStoreHelper」

<http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.websphere.java.doc.doc/web/apidocs/com/ibm/websphere/rsadapter/GenericDataStoreHelper.html>

WASV8では、DataStoreHelperのエラーマッピングに定義された例外で置換する/しないを設定することが出来ます。DB2の場合、例外マッピングモデルの使用を選択した場合(デフォルト値)、com.ibm.websphere.ce.cm.ConnectionWaitTimeoutExceptionサブクラスのSQL例外が発行されます。例外検査モデルの使用を選択した場合、java.sql.SQLTransientConnectionExceptionクラスのSQL例外とcom.ibm.websphere.ce.cm.ConnectionWaitTimeoutExceptionクラスのチェーン例外が発行されます。SQLTransientConnectionExceptionでしか例外をcatchしない等の特別なアプリケーション要件がなければ、デフォルトの設定値で宜しいかと思えます。

Designing Web System Infrastructure with WAS V8.0

### WAS JDBC Provider Type と DataStoreHelper クラスの対応関係 =DB2 の場合=

- DataStoreHelperクラスは、JDBC3.0(db2jcc.jar)を使う場合も JDBC4.0(db2jcc4.jar)を使う場合でも、同じDB2UniversalDataStoreHelperクラスが利用されます。
- DB2, Oracleなど主要なDBMSを使う場合には、JDBCプロバイダー・タイプを選択すると自動的に対応するDataStoreHelperクラスが選択されますので、変更は不要です。
  - ◆ 他DBMSを使うケースや、カスタマイズ済みDataStoreHelperクラスを使いたい場合に明示的にクラス名を指定

DataStoreHelper Class	JDBC Provider Type
com.ibm.websphere.rsadapter. DB2DataStoreHelper.java	DB2 Legacy CLI-based Type 2 JDBC Drivers (*サポート対象外)
com.ibm.websphere.rsadapter. DB2UniversalDataStoreHelper.java	DB2 Universal JDBC Driver Provider
	DB2 Using IBM JCC Driver

© 2012 IBM Corporation 97

DB2のバージョンによってJDBCドライバーの名称、対応するJDBC標準のバージョン、ファイル名などが異なるのは前ページの一覧のとおりですが、

どのJDBCドライバーを使う場合も、DataStoreHelperクラスの選択肢はDB2UniversalDataStoreHelperクラスのみです。

- データ・ソースの設定Overview
- Trusted Context 設定方法
- 接続検証プロパティ
- 監査ログ取得例
- DB2 Client Rerouteの動き
- 接続処理による負荷の軽減(サージ保護)
- DBサーバー過負荷状態におけるリクエストの抑制(滞留タイマー)
- PreparedStatementCacheについて
- 拡張DB2データ・ソース

## 2. WAS-DB2接続設計

### ・WAS-DB2接続設計

- JDBCドライバー設計
  - データ・ソース設計
  - アプリケーション設計
  - パフォーマンス / 問題判別
  - Hints & Tips
- ・WAS-Oracle接続設計

## データ・ソースの設定

Designing Web System Infrastructure with WAS V8.0

■ 管理コンソール

データ・ソース作成時に選択した値が、そのまま反映される。その他は、デフォルト値が設定される。

リソースの有効範囲とテスト接続時に実行されるJVMの関係  
 セル            DeploymentManager  
 クラスター    NodeAgent  
 ノード         NodeAgent  
 サーバー       AppServer

認証情報は、JAAS - J2C認証データにて登録する。

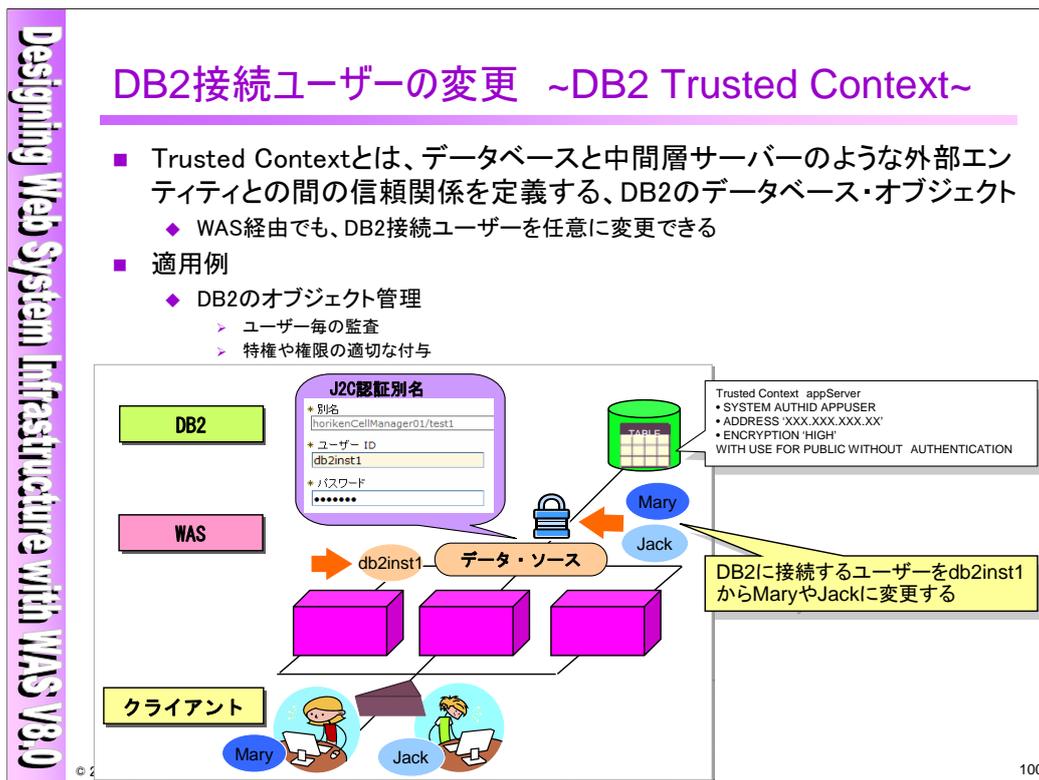
接続先のデータベース情報を登録する。

© 2012 IBM Corp. 99

データ・ソースは、以下のリンク先などをご参考にされ、管理コンソールからデータ・ソース名、JDBCプロバイダーの選択、データ・ストアのヘルパー・クラス、セキュリティなどを設定し、新規作成して下さい。設定後の管理コンソールの画面が上記スライド部分になります。

・WASV8.0 Information Center - 「DB2 用データ・ソースの最小必要設定」

[http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/topic/com.ibm.websphere.nd.multipatform.doc/info/ae/ae/rdat\\_minreqdb2dist.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/topic/com.ibm.websphere.nd.multipatform.doc/info/ae/ae/rdat_minreqdb2dist.html)



DB2にアクセスするアプリケーション・サーバーの多くは、通常1つのIDによりデータベース・アクセスを行います。

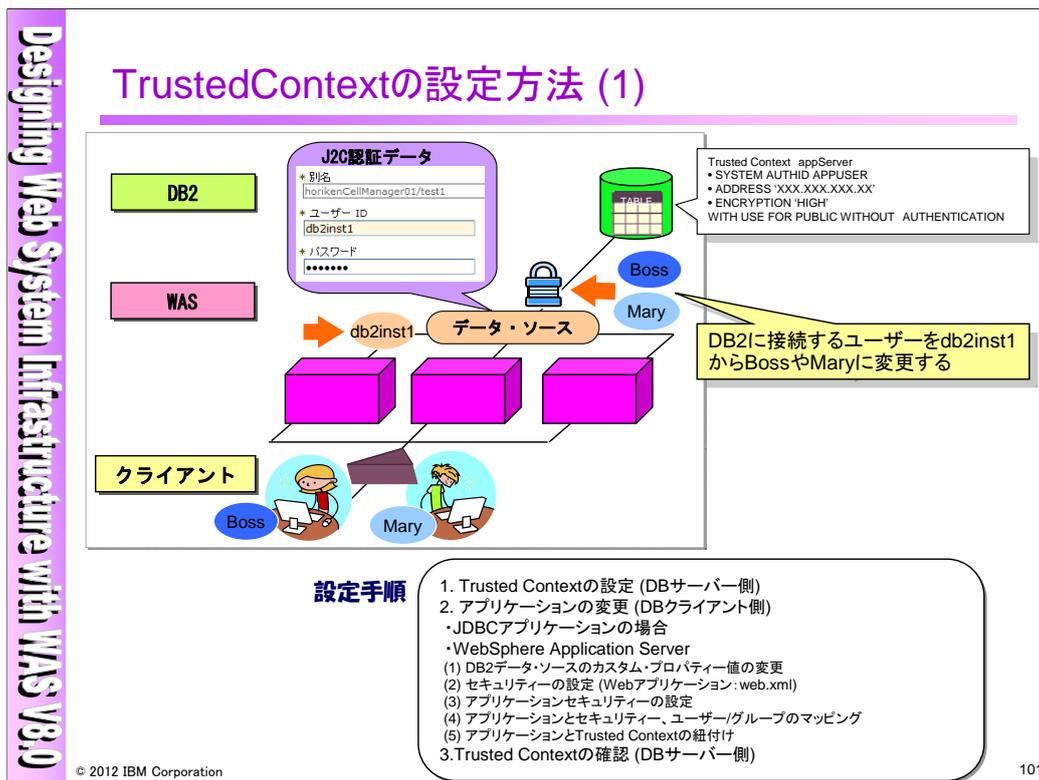
このアプローチには、多くの運用上の利点があります。DB2とのすべてのやり取りが1つのユーザーIDによって管理されるので、アクセス制御が容易になりますが、これにより、どのユーザーも一律に同じデータにアクセス可能な状態となることに加え、誰がどの表にアクセスしたのかわからなくなります。

Trusted Contextとは、DB2データベースと中間層サーバーのような外部エンティティ(アプリケーション・サーバーなど)との間の信頼関係を定義するデータベース・オブジェクトです。この機能を使用することにより、WAS経由でのDB2アクセスでも、データ・ソースやgetConnectionの引数に設定したユーザーではなく、任意のユーザーでDB2に接続することが可能になります。これにより、ユーザー毎に監査することができます。また、ユーザー毎にテーブルのアクセス権限等を付与することが可能になります。

Trusted Contextの前提条件は以下になります。また、WASV8以降では、WASV6.1と比較すると管理コンソールでの設定方法が簡便化されています。設定手順につきましては、「DB接続設計-参考-」のP.7~P.13をご確認下さい。

- DB2 v9.5以降 (AIX, HP-UX, Linux, Solaris, Windows)
- DB2 v9.1以降 (z/OS)
- WAS v6.1.0.11以降

相次ぐ不祥事やコンプライアンスの欠如を防止するため、日本の法規制(日本版SOX法)が整備されました。この法規制では、自社の財務報告に不正や誤りが生じないよう監視やチェックの体制を築くことが求められており、Trusted Contextはそれを(一部)実現できる機能となりますので、ご検討下さい。



Trusted Contextとは、データベースと中間層サーバーのような外部エンティティ(アプリケーション・サーバーなど)との間の信頼関係を定義するデータベース・オブジェクトです。Trusted Contextを利用するには、WAS V6.1.0.11以降、DB2 V9.5以降(AIX、HP-UX、Linux、Solaris、Windows)、DB2 V9.1以降(z/OS)を使用する必要があります。

**前提条件**

- DB2 v9.5以降 (AIX, HP-UX, Linux, Solaris, Windows)
- DB2 v9.1以降 (z/OS)
- WAS v6.1.0.11以降

Designing Web System Infrastructure with WAS V8.0

## TrustedContextの設定方法 (2)

- 1. Trusted Contextの設定 (DBサーバー側)
  - ◆ Trusted Contextの設定

```

$ db2 -tvf trust.sql
connect to sourcedb user secadmin using
  Database Connection Information
  Database server      = DB2/AIX64 9.5.0
  SQL authorization ID = SECADMIN
  Local database alias = SOURCEDB
drop trusted context my_db2_tcx
DB20000I The SQL command completed successfully.
create trusted context my_db2_tcx based upon connection using system authid db2inst1
attributes (address 'x.xxx.xxx.xx') with use for Mary with authentication,
public without authentication enable
DB20000I The SQL command completed successfully.

```

db2inst1は、最初のTrusted connectionをx.xxx.xxx.xxから接続

一度確立したTrusted connectionにおいて、Mary以外のすべてのユーザーは、認証なしにユーザーIDのみで接続を再利用できる

© 2012 IBM Corporation 102

上記スライド部分では、DBサーバー側においてTrustedContextの設定を行います。

ユーザー:db2inst1は、最初のTrusted connectionをIPアドレス(x.xxx.xxx.xx)から接続を行います。  
また、一度確立したTrusted connectionにおいて、Mary以外のすべてのユーザーは、認証なしにユーザーIDのみで接続を再利用できる設定です。

Designing Web System Infrastructure with WAS V8.0

## TrustedContextの設定方法 (3)

- 2. アプリケーションの変更 (DBクライアント側)
  - ・JDBCアプリケーションの場合  
(<DBROOT>/samples/java/jdbc/TrustedContext.java)

```

// Universal JDBC Driver using DataSource
DB2ConnectionPoolDataSource db2ds = new DB2ConnectionPoolDataSource();
db2ds.setDatabaseName("xxDB");
db2ds.setServerName ("x.xxx.xxx.xx");
db2ds.setDriverType (4);
db2ds.setPortNumber(yyyyy);
java.util.Properties properties = new java.util.Properties();
String user = "db2inst1";
String password = "db2inst1";
Object[] objects = db2ds.getDB2TrustedPooledConnection(user,password,properties);

【省略】
PooledConnection pooledCon = (PooledConnection)objects[0];
byte[] cookie = ((byte[])(objects[1]));
BufferedReader input = new BufferedReader(new InputStreamReader(System.in));
System.out.print("Input switch user name : ");
String newuser = input.readLine();
System.out.print("Password : ");
String newpassword = input.readLine();
String userRegistry = null;
byte[] userSecTkn = null;
String originalUser = "db2inst1";
Connection con =((DB2PooledConnection)pooledCon).getDB2Connection(
cookie,newuser,newpassword,userRegistry,userSecTkn,originalUser,properties);

```

© 2012 IBM Corporation 103

上記スライド部分では、DBクライアント側においてTrustedContextに対応するためのアプリケーションの変更を行います。

上記は、JDBCアプリケーションの場合の例になり、最初にユーザー: db2inst1にてTrusted connectionを確立し、任意のユーザーにスイッチしてdb2inst1の接続を再利用しています。

Designing Web System Infrastructure with WAS V8.0

## TrustedContextの設定方法 (4)

- 2. アプリケーションの変更 (DBクライアント側)
  - ◆ (1) DB2データ・ソースのカスタム・プロパティ値の変更

<code>propagateClientIdentityUsingTrustedContext</code>	<code>true</code>	Enable the propagation of client identity using DB2 Trusted Context. The resource reference must be configured with trusted connections authentication for this setting to take affect.	<code>false</code>
---	-------------------	---	--------------------

trueに設定する。デフォルトはfalse。

- ◆ (2) セキュリティーの設定 (Webアプリケーション: web.xml)

保護するリソースを設定する

© 2012 IBM Corporation 104

上記スライド部分では、WAS経由のアプリケーションの場合の例になり、WebSphere側での設定変更が必要になります。

データ・ソースのカスタム・プロパティを設定し、該当のアプリケーションとTrustedContextの設定を紐付けて下さい。

Designing Web System Infrastructure with WAS V8.0

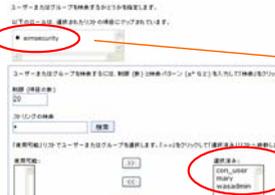
## TrustedContextの設定方法 (5)

- 2. アプリケーションの変更 (DBクライアント側)
  - ◆ (3) アプリケーション・セキュリティの設定
    - ▶ 管理コンソールより、セキュリティ > 管理、アプリケーション、およびインフラストラクチャーの保護を選択する



管理セキュリティ、アプリケーションセキュリティを設定する

- ◆ (4) アプリケーションとセキュリティ、ユーザー/グループのマッピング
  - ▶ 管理コンソールより、エンタープライズ・アプリケーション > 該当のアプリケーション > ユーザー/グループ・マッピングへのセキュリティ・ロールを選択する



web.xmlに指定したセキュリティ・ロールが表示される

switchするユーザー情報を登録する

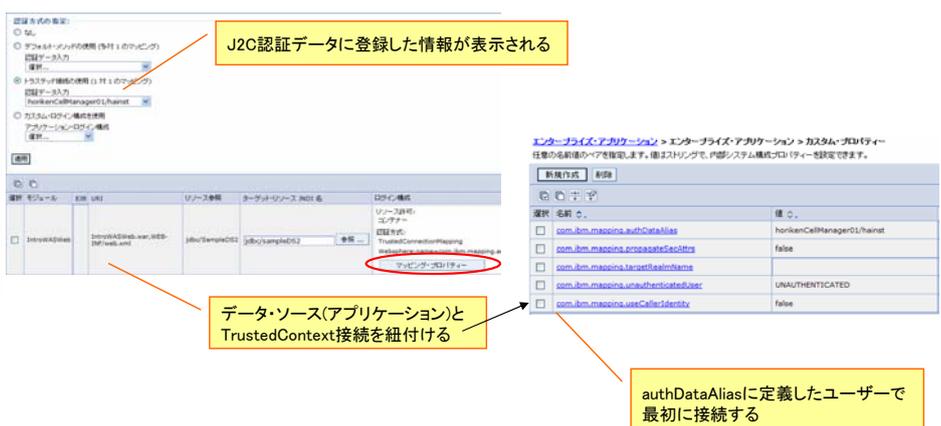
© 2012 IBM Corporation 105

上記スライド部分を参考にして設定して下さい。一般的なアプリケーション・セキュリティと同様です。

Designing Web System Infrastructure with WAS V8.0

## TrustedContextの設定方法 (6)

- 2. アプリケーションの変更 (DBクライアント側)
  - ◆ (5) アプリケーションとTrusted Contextの紐付け
    - ▶ 管理コンソールより、エンタープライズ・アプリケーション >- 該当のアプリケーション >- 参照 >- リソース参照を選択する



© 2012 IBM Corporation

上記スライド部分では、アプリケーションとTrusted Contextを紐付けています。

Designing Web System Infrastructure with WAS V8.0

## TrustedContextの設定方法 (7)

- 3. Trusted Contextの確認 (DBサーバー側)
  - ◆ db2 list applicationsコマンド

Auth Id	Application	Appl.Handle	Application Id	DB Name	# ofAgents
Mary	db2jcc_applica	2935	9.188.198.118.37939.08040804381	SOURCEDB	1

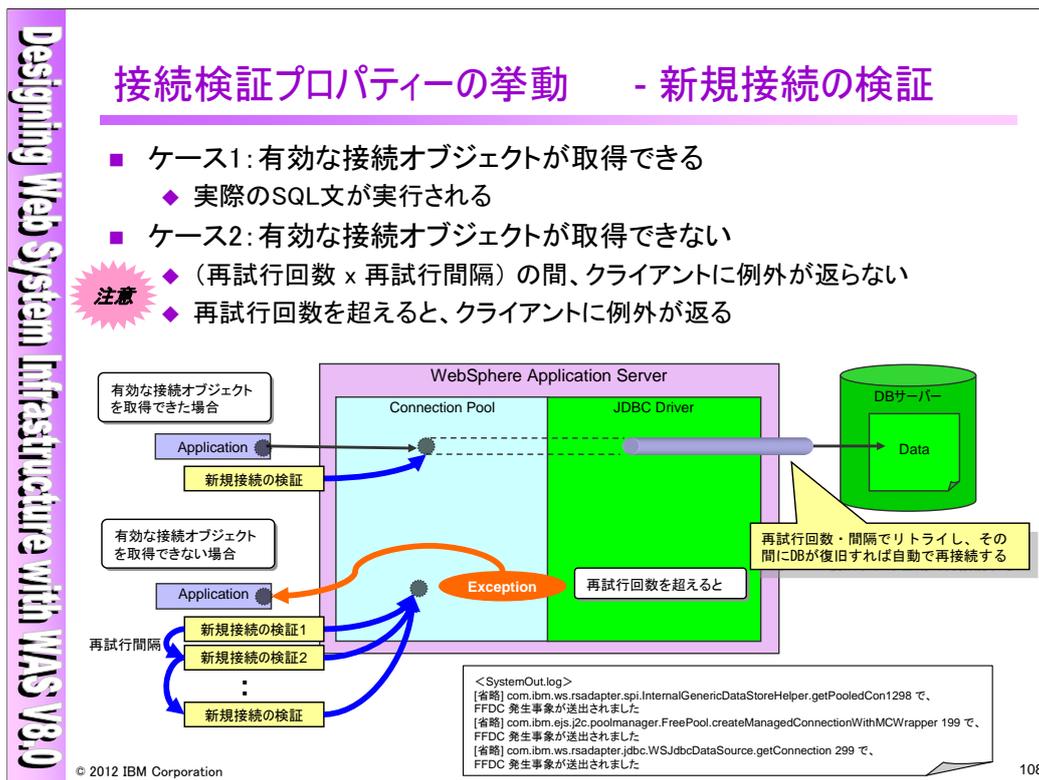
db2inst1がMaryにスイッチし、db2inst1の接続を再利用

- ◆ db2pd -applications -d sourcedbコマンド

Database Partition 0 -- Database SOURCEDB -- Active -- Up 0 days 00:17:03									
Applications:									
Address	AppHandle	[nod-index]	NumAgents	CoorEDUID	Status	C-AnchID	C-StmtUID	L-	
AnchID	L-StmtUID	Appid				WorkloadID			
WorkloadOccID									
0x0780000000FC9200	2935	[000-02935]	1	10232	UOW-Waiting	0	0		57
1	9.188.198.51.50253.071120074513					1	3		
External Connection Attributes									
Address	AppHandle	[nod-index]	ClientIPAddress	EncryptionLvl	SystemAuthID				
0x0780000000FC9200	2935	[000-02935]	9.188.198.51	None	Mary				
Trusted Connection Attributes									
Address	AppHandle	[nod-index]	TrustedContext	ConnTrustType	RoleInherited				
0x0780000000FC9200	2935	[000-02935]	MY_DB2_TCX	explicit trusted connection	n/a				

© 2012 IBM Corporation 107

ここでは、DBサーバー側においてTrustedContextの設定確認を行っています。db2 list application コマンドでは、スイッチ後のユーザー情報を確認できます。db2pd -applications -d sourcedbコマンドでは、スイッチ後のユーザー情報とTrusted Contextを使用していること(explicit trusted connection)が確認できます。

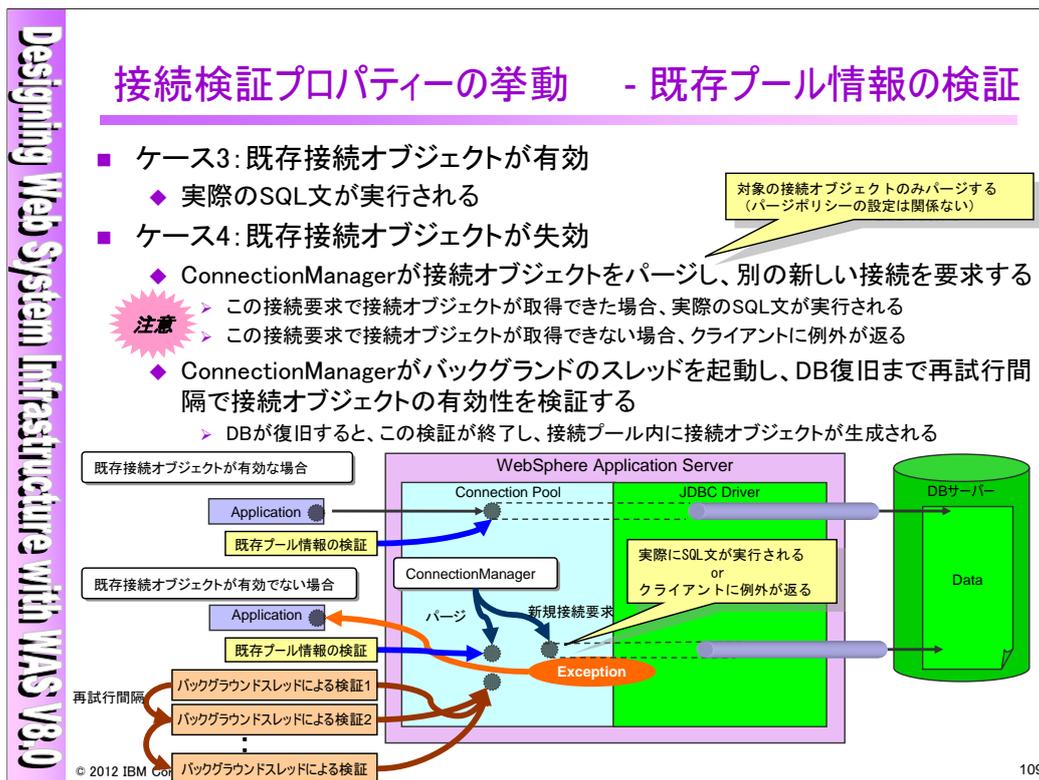


新規接続の検証では、isValid()メソッドの実行により、有効な接続オブジェクトが取得できると実際のSQL文が実行されます。反対に、有効な接続オブジェクトが取得できない場合、再試行回数 × 再試行間隔の間、クライアントに例外は返りません。ブラウザの画面は実行中のままになりますので、ご注意ください。そして、再試行回数を超えると、クライアントに例外が返ります。

クライアントに例外を返さずにDBへの自動再接続を何秒間(再試行回数 × 再試行間隔)試みてよいのか、というシステム要件をご確認の上、設定して下さい。

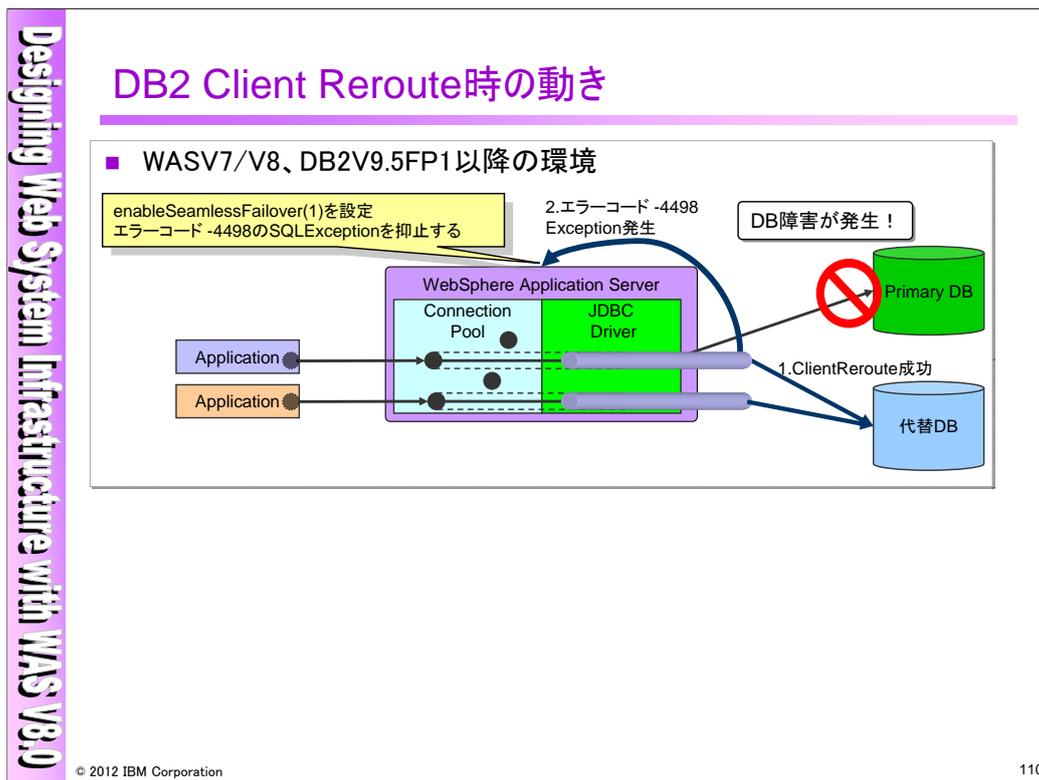
考え方としては、DBサーバーがコールド・スタンバイ構成であるなら、DBサーバーのテークオーバーが完了するまでにかかる時間より少し長い時間にする必要があります。ただし、あまり長いとDB障害時にそれだけの時間クライアントが待たされることとなります。

デフォルト値の再試行回数:100/再試行間隔:3ですと、DB障害時に300秒間クライアントに例外が返りません。この設定値は、若干大きいかと思しますので、システム特性をご考慮の上、チューニングして下さい。



既存プール情報の検証では、isValid()メソッドの実行により、接続プールに蓄積されている接続オブジェクトが有効であると実際のSQL文が実行されます。反対に、接続オブジェクトが有効ではない場合、ConnectionManagerが該当の接続オブジェクトをパージし、別の新しい接続を要求します。この接続要求で接続オブジェクトが取得できた場合、実際のSQL文が実行されます。この接続要求で接続オブジェクトが取得できない場合、クライアントに例外が返ります。そして、Connection Managerがバックグラウンドのスレッドを起動し、DBが復旧するまで再試行間隔で接続オブジェクトの有効性を検証します。従いまして、再試行間隔は、数秒～数十秒を初期値とされるのが宜しいかと思えます。

また、この「バックグラウンドのスレッドを起動しDBが復旧するまで接続オブジェクトの有効性を検証する機能」が、アプリケーションでStaleConnectionException発生をCatchする実装をした場合との違いになります。アプリケーション実装では、新規接続/既存接続に関わらず、アプリケーション内で指定した回数 x 間隔でDBへの自動再起動を試みます。



WASV7/V8、DB2V9.5の環境では、PrimaryDBに障害が発生すると、DB2ClientRerouteの機能により代替DBにリクエストが割り振られますが、それと同時にエラーコード-4498がクライアントに返ります。このエラーコードはStaleConnectionExceptionにマッピングされているため、接続プール内にプールされている接続オブジェクトがパージされます。アプリケーション側で特に意識することなく、代替DBにアクセスすることができますが、接続プールに蓄積されている接続オブジェクトを再利用できず、新規接続となってしまいます。これは、StaleConnectionExceptionをCatchしている実装と同じ挙動になり、DB2ClientRerouteを使用するメリットが薄れてしまいます。

それに対して、WASV8、DB2V9.5FP1環境では、データソースのカスタムプロパティにenableSeamlessFailover(1)を設定することで、エラーコード-4498のSQLExceptionを抑制でき、代替DBへ接続プールに蓄積されている接続オブジェクトで接続することができます。

PrimaryDBと代替DBで、IPアドレスの付け替えを行う構成でも、DB2ClientReroute機能を使用するメリットがあると言えます。

・DB2ClientReroute成功時のSystemOut.logへの出力メッセージ

```
[09/03/25 17:54:21:859 GMT+09:00] 0000002b SystemErr    R
com.ibm.websphere.ce.cm.StaleConnectionException: [jcc][t4][2027][11212][4.2.73] 接続は失敗しましたが、再確立されました。ホスト名または IP アドレスは "xxx.makuhari.japan.ibm.com" で、サービス名またはポート番号は xx,xxx です。
```

特殊レジスターのやり直しは可なことも不可なこともあります (理由コード = 1)。ERRORCODE=-4498, SQLSTATE=08506

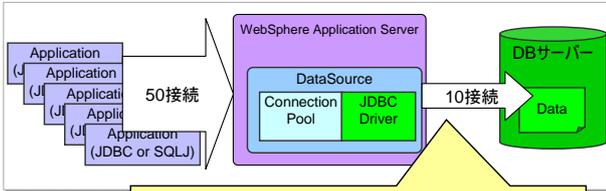
・クライアント・リルート機能の補足

“クライアント転送サーバー・リスト JNDI名”には、代替DBに接続するデータソースを定義し、そのデータソース名を設定する。この設定により、Primary DB障害時には、新規接続はPrimary DB→代替DBではなく代替DBに対して接続する動きとなる。この時、namestore.xmlファイルをクライアント転送サーバー・リストJNDI名で設定した値に書き換える。さらに、“JNDI からのアンバインド・クライアント転送リスト”を設定すると、代替DB→PrimaryDBの接続順をリセットすることができる。

Designing Web System Infrastructure with WAS V8.0

## 接続処理による負荷の軽減

- サージ保護設計
  - ◆ DB接続要求数がサージしきい値を超えるとサージ・モードが開始され、サージ作成間隔毎にDBへ接続が要求される
- 選択指針
  - ◆ DBサーバーに同時に大量の接続要求が行われ、接続処理効率が悪化している場合
- 設定例
  - ◆ サージしきい値            10接続
  - ◆ サージ作成間隔            5秒間



ここで、 $50 - 40 = 10$ 接続が待ち状態となる。  
そして、5秒後に11個目の接続が作成される。



© 2012 IBM Corporation
111

データベースに対し、大量の接続要求が同時に行われると、接続処理の効率が悪化する可能性があります。この事象を防ぐためのパラメーターとしてサージ保護があり、拡張接続プール・プロパティにて設定することが出来ます。

例えば、サージしきい値を10接続、サージ作成間隔を5秒間と設定した場合、大量の接続要求があっても同時に10接続でしか接続処理を行いません。その後の要求に対しては、5秒おきに接続処理が行われます。

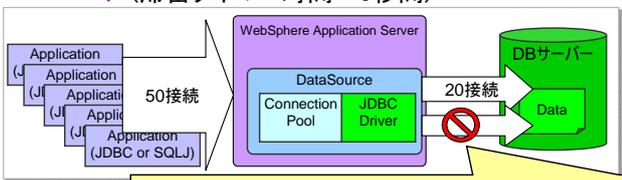
Designing Web System Infrastructure with WAS V8.0

## DBサーバー過負荷状態におけるリクエストの抑制

- 滞留タイマー設計
  - ◆ 滞留時間を経過した接続数が滞留しきい値を超えると滞留状態と判断され、次のリクエスト時にExceptionが返る
- 選択指針
  - ◆ DBサーバーが過負荷に陥っており、更に処理を要求すると、全てのリクエストの処理効率が低下すると想定される場合

**注意** ◆ ResourceExceptionをCatchする仕組みが必要

- 設定例
  - ◆ 滞留時間 10秒間
  - ◆ 滞留しきい値 20個
  - ◆ (滞留タイマー時間) 5秒間



デフォルト値は0 (=off)

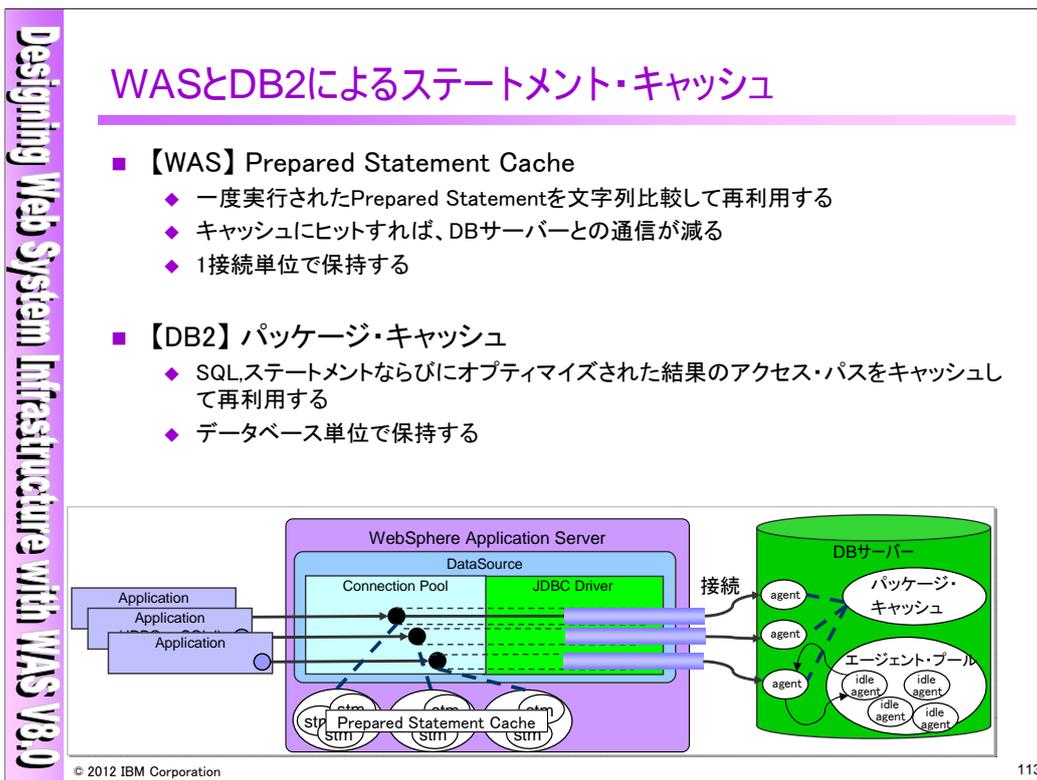
10秒接続された状態の接続が20個を超えると滞留状態となる。  
そして、21個目の接続はResourceExceptionが返る。

過負荷に陥っているデータベースに対して、更に処理を要求すると、全てのリクエストの処理効率が低下する可能性があります。これは最大接続数が大きすぎる、DBサーバーのリソースを占有してしまうような大量検索処理が実行された等が根本的な原因であると考えられますが、この事象による影響を軽減させるためのパラメーターとして滞留モード設計があり、拡張接続プール・プロパティにて設定することが出来ます。

滞留状態でgetConnection()を呼び出すと、javax.resource.ResourceExceptionが発生します。アプリケーションで、この例外を受け取ったら、データベース負荷を回避する為の対応を行うことで、効率的にサービスを提供できます。例えば、画面に「しばらくお待ち下さい」などのメッセージを表示して、ユーザーに待機を促すことで、WASサーバー、しいてはDBサーバーに対する負荷を軽減させることができます。

滞留タイマー時間は、滞留状態をチェックする間隔を設定します。滞留時間はリソースからレスポンスが戻るまでの待ち時間を設定します。この時間を超えると、その接続は滞留接続であると判定されます。滞留しきい値は、接続プールは滞留状態と判定される滞留接続数を設定します。以下を参考し、初期値を設定して下さい。

滞留しきい値 < DataSource最大接続プールサイズ < WebContainerスレッドプール  
0 < 滞留時間 < 接続タイムアウト



この前の接続プール設定では、接続オブジェクトをプールし再利用する技術についてご紹介しました。

こちらのページでは、接続以外にWASやDB2が再利用を目的としてキャッシュする、ステートメント・キャッシュと、パッケージ・キャッシュについてご説明します。

WASのステートメント・キャッシュは、一度実行されてPrepared Statementを文字列で比較して再利用します。キャッシュにヒットすれば、DBサーバーとの通信が減り、パフォーマンスの向上が見込めます。また、Prepared Statementによる実行が必要です。

DB2のパッケージ・キャッシュは、動的SQLのアクセスパス、静的SQLのパッケージをキャッシュします。データベース共有メモリー上で確保され、DB2が使用するメモリーとしてはdb2agentを除けばバッファ・プールに次いで大きな領域になります。DB単位でキャッシュを保持します。

なお、キャッシュがクリアされるタイミングは、下記の通りです。

【WAS】Prepared Statement Cache ⇒ Connectionオブジェクトに登録されているため、接続パーージもしくはJVM再起動によりクリアされる

【DB2】パッケージ・キャッシュ ⇒ データベースオブジェクトの定義変更に伴って、該当オブジェクトのキャッシュがクリアされる

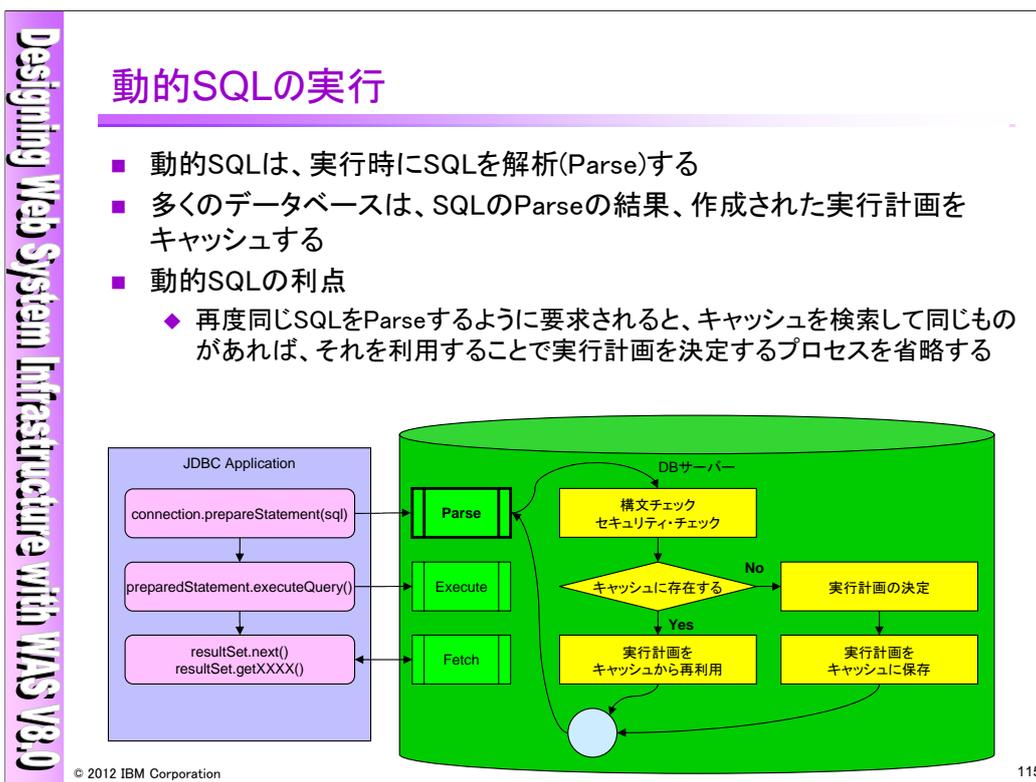
Designing Web System Infrastructure with WAS V8.0

## 動的SQLと静的SQL

- 動的SQL (JDBC)
  - ◆ 実行前
    - Compile
  - ◆ 実行時
    - Parse / Execute / Fetch
- 静的SQL (SQLJ)
  - ◆ 実行前
    - Precompile / Compile / Bind
  - ◆ 実行時
    - Execute / Fetch
- 考慮点
  - ◆ どちらの方法も、JDBCドライバーを利用してデータベースに接続する
  - ◆ SQLJでは事前にBindしておくことで、実行時にParseを行わず、高速にSQLを実行
  - ◆ SQLJのソースが変更になるたびにBindを行う必要がある
  - ◆ アプリケーションのバージョン管理が、SQLJの方が複雑である
  - ◆ アプリケーション開発者は、SQLJ固有の構文を理解する必要がある

© 2012 IBM Corporation 114

動的SQLと静的SQLでは、静的SQLが事前に解析処理を行っている分、高速にSQLの実行が可能です。ただし、アプリケーションのバージョン管理では、serファイルやクラスファイルの管理、また事前にBindする必要があるため、動的SQLに比べ複雑になります。また、静的SQLではSQLJの構文の知識も必要となります。



動的SQLとは、実行時にSQLを解析(Parse)する方法です。解析結果はRDBMSにキャッシュされ再利用されます。

Designing Web System Infrastructure with WAS V8.0

## 静的SQLの実行

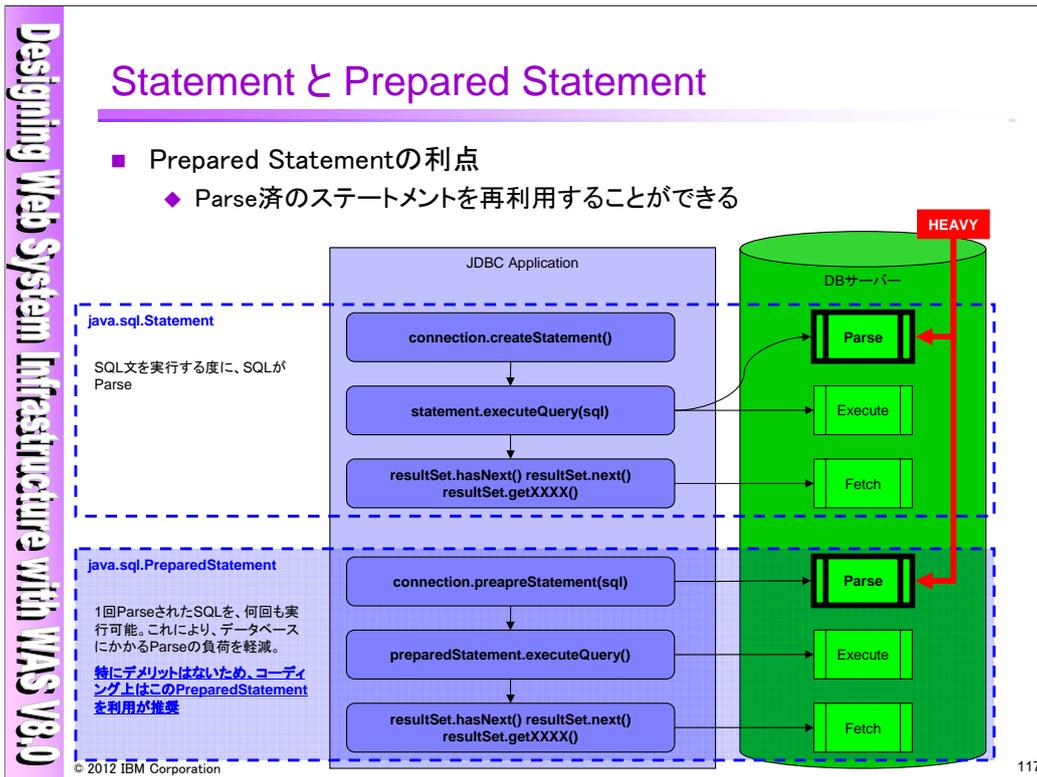
- SQLJを利用すると、静的SQLが利用可能になる
- 静的SQLの利点
  - ◆ SQLに関する情報を事前にBindすることで、実行時にSQLをParseする必要がない

### DB2におけるSQLJ開発フロー

- 開発者は、SQLJのアプリケーション・コード(\*.sqlj)を記述する
- データベース・ベンダーが提供するSQLJトランスレーター(DB2 UDBの場合sqljコマンド)を利用して、プリコンパイルすると、Javaアプリケーション・コードと、SQLJプロファイル(DB2 UDBの場合、\*.ser形式)が生成される
- データベース・ベンダーが提供するSQLJプロファイルBinder(DB2 UDBの場合db2sqljcustomizeコマンドやdb2sqljbindコマンド)を利用して、BINDする。このとき\*.serファイルには、BIND情報が更新される
- Javaアプリケーション・コードは、通常通りコンパイルしてできた\*.classファイルを、配布する。また、SQLJプロファイルは実行時にも必要であり、\*.classと一緒に配布する。

SQLJを利用すると静的SQLを利用することが可能になり、解析作業を事前にBindしておくことで実行時の解析処理を省略できます。

DB2におけるSQLJアプリケーションの開発フローは、上記スライド部分をご確認下さい。SQLJのファイルをpre-compileすることでjavaファイルおよびSQLJ profileのserファイルが作成されます。javaファイルはそのままcompileし、通常のclassとして使用します。profile customizerコマンドによりpackageの名前などを設定し、その情報をserファイルにも反映します。その後、serファイルに含まれるpackage情報とSQLJ profile binderを使用してデータベースにBindします。



動的SQLの実行には、java.sql.Statementクラスを使用する場合と、java.sql.preparedStatementクラスを使用する場合があります。

java.sql.Statementを使用した場合は、ParseとExecuteが同時に実行されます。

java.sql.preparedStatementを使用した場合は、ParseとExecute処理を分離することができます。一度ParseしたSQLはキャッシュされ再度利用する事が可能となります。特にデメリットはありませんので、基本的には、PreparedStatementを使用して下さい。(PreparedStatement Cacheを使用する際の前提条件となります。)

Designing Web System Infrastructure with WAS V8.0

## Prepared Statement Cache設定

- 仕組み
 

1. アプリケーションにてSQL文発行
  2. データベースがプリコンパイル処理
  3. execute直前の状態をWASとDBのメモリー上に保持
  4. 次に同じステートメントが来たときは、3を利用してすぐにexecute処理を実行
- 設定指針
  - ◆ 【WAS】 発行されるSQL数を設定する (1接続単位)
    - java.sql.PreparedStatmentを使用する
    - Cacheの再利用条件に注意する
    - TPVIによるチューニングを実施する
  - ◆ 【DB2】 パッケージ・キャッシュはデフォルトでは自動調整されるため、基本的には考慮する必要はない

Prepared Statement Cacheについての詳細は、巻末資料「DB接続設計-参考-」をご参照下さい。

© 2012 IBM Corporation
118

JDBCは動的なSQLのインターフェースのため、SQL発行→SQL文のプリコンパイル (prepare) → 実行 (execute) というプロセスが、同じSQLが発行されたとしてもリクエスト毎に実行されます。データベースにとってはprepare処理は負荷が大きいので、prepare処理の実行回数を軽減させることによりパフォーマンスが向上します。そこで、アプリケーションから要求されたSQL文をデータベースがプリコンパイル処理し、executeする直前の状態をWASとDBのメモリー上にexecuteが終了した後も常時保持しておき(※)、次に同じステートメントが来た時に再利用するようにした仕組みが、Prepared Statementを用いたSQLの処理です。

(※)SQL文とそのSQL文に対するステートメントとの紐付けを行ない、コネクションに保持します。

Prepared Statement Cacheを利用する際のチューニング・ポイントは以下になります。

WAS側: 1接続あたり、いくつのステートメントを保持するかを設定して下さい。可能であれば、発行されるSQL数を設定することが望ましいです。

DB側: WAS側の設定を考慮して、CLIPKG(パッケージ数)とdb2agentのヒープをチューニングして下さい。

Designing Web System Infrastructure with WAS V8.0

## Prepared Statement Cacheの考慮点 (1)

- WAS側の考慮点
  - ◆ java.sql.PreparedStatementを使用する
 

java.sql.statement	SQL文が実行される度に、SQLがParseされる
java.sql.PrepareStatement	1回parseされたSQLを何回も実行可能
  - ◆ Prepared Statement Cacheの再利用条件
    - SQLの文字列が全く同じであること
      - 1文字でも異なると判断されると、再利用されない
      - 列名指定順、大文字と小文字、空白文字数が違うと、異なるステートメントと判断される
    - preparedStatement()で指定する引数が同じであること
      - resultSetType, resultSetConcurrency, resultSetHoldability
  - ◆ Tivoli Performance Viewer (TPV)による監視
    - TPVにて破棄されたPrepared Statement数を確認する
      - PMIモニター・レベルを”拡張”、もしくは”カスタム”にて定義する

<input type="checkbox"/>	PrepStmntCacheDiscardCount	CountStatistic	キャッシュが選択のために破棄されるステートメントの数。	使用不可
--------------------------	----------------------------	----------------	-----------------------------	------

- この数が極端に多い場合には、PrePared Statement Cacheを大きくすることを検討する
- Statement Cacheが利用するHeapの量はそれほど多くないが、変更後は負荷テストを行う

119

© 2012 IBM Corporation

ステートメント・キャッシュを再利用するには、java.sql.PreparedStatementを使用して下さい。また、SQLの文字列が全く同じである必要があります。列名の指定順の違いや、大文字、小文字の違い、空白文字数の違い、またpreparedStatement()の引数が異なる場合などはキャッシュを利用することが出来ません。

Tivoli Performance Viewerなどのリソースの利用状況を監視するツールを使用することで、システム稼動中に接続が持っているステートメント数と廃棄状況が分かりますので、その情報を元にステートメントの数をチューニングして下さい。

Designing Web System Infrastructure with WAS V8.0

## Prepared Statement Cacheの考慮点 (2)

- DB2側の考慮点
  - ◆ CLIPKG (パッケージ数)
    - ▶ WAS側の設定に応じて、DB2側もCLIPKGをチューニングする

<パッケージ数が足りなくなった場合のエラーメッセージ>  
 com.ibm.db2.jcc.b.SqlException: パッケージNULLID.SYSLN3030X5359534C564C3031"が見つかりませんでした。
  - ◆ db2agentのヒープ・サイズ (applheapsz)
    - ▶ デフォルト設定値256(約1MB)は一般的に小さいため、チューニング初期値として、512・1024程度に設定する
    - ▶ APPLHEAPSZモニター (db2agentプロセス毎)
    - ▶ ヒープ不足のエラー(SQL0954C)が発生しない場合、徐々に値を小さくする

```
#db2 get snapshot for applications for <DB名>
アプリケーション・スナップショット
アプリケーション・ハンドル      = 79
アプリケーション状況            = 接続完了
      :
      :
メモリ・プール・タイプ          = アプリケーション・ヒープ
  現行サイズ (バイト)            = 65536
  最高水準点 (バイト)            = 65536
  構成済みサイズ (バイト)        = 1245184
```

```
# db2mtrk -p
トラッキング・メモリ: 2009/02/11 / 15:16:19
      apph  other  apcctlh
      64.0K  64.0K  64.0K
```

© 2012 IBM Corporation 120

DB2V9.5のCLIPKGのデフォルト値は1344ステートメントですので、WAS側のPrepared Statement Cacheの設定を考慮し、設定して下さい。パッケージ数が足りなくなった場合には、スライド部分のSQLExceptionが発生します。

db2agentのヒープサイズは、環境に大きく依存しますが、チューニング初期値として512/1024程度を設定して下さい。このヒープサイズはアプリケーションスナップショット、もしくはdb2mtrkコマンドにて確認できます。

Designing Web System Infrastructure with WAS V8.0

## 拡張DB2データ・ソース(1/2)

- アプリケーション(EAR)毎に異なるデータ・ソースのカスタム・プロパティを設定できる機能
  - ◆ 接続プールの共用が可能
    - ▶ WAS / DB2のメモリー消費を抑える
    - ▶ トランザクション内でのコネクション共用により、パフォーマンスが向上

**WASV6.xの構成**

(例)DB2のスキーマ名がアプリ1とアプリ2で異なるため、データ・ソースを別々に定義した

**WASV8の構成**

プロパティ値をアプリケーション単位で設定できる。コネクション・プールを共用できる

- 前提条件
  - ◆ DB2 Universal JDBC driver Version 4.3.81 or higher
  - ◆ DB2 Using IBM JCC driver Version 3.53.65 or higher

© 2012 IBM Corporation 121

拡張DB2データ・ソースとは、アプリケーション(EAR)毎に異なるデータ・ソースのカスタム・プロパティを設定できる機能です。この機能を有効活用することで、異なるデータ・ソース・プロパティ値を設定しても、コネクション・プールを共用することができ、WAS/DB2サーバーでメモリー消費を抑えることができます。また、トランザクション内でのコネクションが共用され、パフォーマンスを向上されることもできます。反対に、異種プーリング環境にて取得/使用/クローズ/接続パターンを最適化し2フェーズ・コミット処理を実施したくない場合には、管理コンソールにて「異機種混合プールで取得/使用/クローズ/接続パターンの最適化」にチェックをして下さい。

この拡張DB2データ・ソースは、DB2 Universal JDBC DriverとDB2 Using IBM JCC Driverをサポートします。

・WAS V8.0 InformationCenter - 「アプリケーション・レベルでの DB2 データ・ソース定義の拡張」  
[http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tdat\\_heteropool.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tdat_heteropool.html)

## 拡張DB2データ・ソース(2/2)

### ■ 設定方法

- ◆ 管理コンソールより、アプリケーション → 参照(リソース参照) → 拡張プロパティを選択する



### ■ 設計指針

- ◆ アプリケーション(EAR)毎に異なるカスタム・プロパティ値を設定し、かつ接続プールを共用したい場合

### ■ 適用例

- ～同一のデータ・ソースを使用し、
  - ◆ アプリケーション毎にDB2のスキーマ名を変更する
  - ◆ アプリケーション毎にJDBCドライバーのログレベルやログ出力先を設定する

© 2012 IBM Corporation

122

拡張DB2データ・ソースは、管理コンソールより、アプリケーション → 参照(リソース参照) → 拡張プロパティにて設定します。アプリケーションが必要なカスタム・プロパティの名前と値のペアを指定します。(例: currentSchema、clientApplicationInformationなど) wasはアプリケーション・サーバーで事前定義されているプロパティ用に予約されているため、プロパティ名の接頭部として使用しないで下さい。

拡張DB2データ・ソースにて、上書きできないプロパティ名は以下になります。

accountingInterval、dataSourceName、databaseName、kerberosServerPrincipal、loginTimeout、logWriter、password、pkList、planName、portNumber、readOnly、securityMechanism、serverName、user

- JNDIによるlookup
- 接続の共用スコープ
- 監査ログ取得例
- JavaEEアプリケーションにおける分離レベルの設定方法
- 分離レベルの確認方法

## 2. WAS-DB2接続設計

### ・WAS-DB2接続設計

- JDBCドライバー設計
  - データソース設計
  - アプリケーション設計
  - パフォーマンス / 問題判別
  - Hints & Tips
- ・WAS-Oracle接続設計

Designing Web System Infrastructure with WAS V8.0

## JNDIによるlookup

- Java Naming and Directory Interface
  - ◆ オブジェクトと名前を関連づけ、その名前をキーとしてオブジェクトを検索する
- 直接参照（非推奨）
  - ◆ アプリケーションから直接lookupする
- リソース参照
  - ◆ 参照名とJNDI名のマッピングをデプロイメント記述子で行う
  - ◆ アプリケーションからは、参照名をlookupする
  - ◆ データ・ソースの認証方法 / 共用スコープ / 分離レベル設定を変更可能

**SystemOut.log抜粋**

J2CA0294W: リソース jdbc/datasource1 を直接 JNDI ルックアップしています。次のデフォルト値が使用されます:

```
[Resource-ref CMConfigData key items]
res-sharing-scope: 0 (SHAREABLE)
res-isolation-level: 0 (TRANSACTION_NONE)
res-auth: 1 (APPLICATION)
isCMP1_x: false (not CMP1.x)
isJMS: false (not JMS)
commitPriority: 0
loginConfigurationName: null
loginConfigProperties: null
Resource ref name: not set
```

参照名とJNDIのマッピングは、「DB接続設計-参考-」のP.23をご参照下さい。

© 2012 IBM Corporation
124

はじめに、アプリケーションから管理コンソールに定義したデータ・ソースを呼び出す仕組みをご説明致します。

JNDIとはJava Naming and Directory Interfaceの略で、Javaでネーミング・サービス、ディレクトリ・サービスを扱うためのインターフェースです。ネーミングサービスとは、オブジェクトと名前とを関連づけ、その名前をキーとしてオブジェクトを検索するためのサービスです。このネーミングサービスの仕組みにより、アプリケーションから外部リソースへのアクセス方法が統一でき、さらに外部リソースに変更が生じた場合にアプリケーションコードを変更することなく、デプロイメント記述子のみの変更で対応できるようになります。つまり、アプリケーションのポータビリティを向上させる仕組みでもあります。JNDI名で取得する対象オブジェクトはデータ・ソースの他にも様々なものがあります。これらのオブジェクト全てにJNDIという統一された仕組みでアクセスできます。

直接参照は、アプリケーション・サーバー上のデータ・ソースを直接lookupする方法です。Java EEでは非推奨となっています。JNDI名を直接アプリケーション・コード内で指定するため、特定の環境情報が記述されてしまうことになり、ポータビリティが低下してしまいます。また、WASV5.0.2.3からは、直接参照を行っている場合には、リソース参照定義は使用されず、デフォルトの設定が使用されます。

リソース参照は、アプリケーションのデプロイメント記述子にリソース参照としてデータ・ソースのポインターを定義します。これを利用することで、デフォルトの分離レベルや、認証方法、共用スコープ等を変更できます。

JNDIクライアントにおいてlookupを行う際は、lookupするオブジェクトの名前の記述フォーマットとして[ java:comp/env/<参照名> ]を使用することが原則であり、Java EE的により作法とされています。アプリケーション・コードで使用する参照名とWASサーバー側で使用するJNDI名のマッピングは、デプロイメント記述子で行います。

Designing Web System Infrastructure with WAS V8.0

## JNDI名と参照名のマッピング

- アセンブリ時
  - Web.xml
  - アプリケーションコード <リソース参照>  
ctx.lookup("java:comp/env/jdbc/ResRef1")
  - アプリケーション・サー  
バー内での参照名 = JNDI名
  - 紐付け
  - アプリケーションコード内での参照名  
= リソース参照名
  - ibm-web-bnd.xml
  - 管理コンソール
- デプロイ時
  - 管理コンソール  
アプリケーションのインストール画面
  - 参照名

© 2012 IBM Corporation 125

アプリケーションのアセンブリ時にJNDI名と参照名をマッピングさせるには、IBM Rational Application Developer (RAD) 等の開発ツールを使用して、Webアプリケーション・デプロイメント記述子に設定します。

また、アプリケーションのデプロイ時にJNDI名と参照名をマッピングさせることも可能です。スライド部分の図は、管理コンソールからアプリケーションをインストールしている途中の画面です。「リソース参照をリソースにマップ」画面でマッピングを行うことができます。

Designing Web System Infrastructure with WAS V8.0

## アノテーションを使用したJNDIによるlookup

- アノテーションとは、ソースコード中にコードの属性情報を付加する機能
  - ◆ アノテーション・サポートによる開発効率・保守効率の向上
    - ▶ ベンダー依存のデプロイメント記述子の排除
    - ▶ 仕様書 : JSR175 Metadata Facility for Java / JSR250 Common Annotation
- 従来の方法
 

```
Context ctx = new InitialContext();
ds = (DataSource) ctx.lookup("java:comp/env/jdbc/ResRef1");
```
- アノテーション(@Resource)を使用したlookup
 

```
@Resource(name="jdbc/ResRef1")
```

デフォルト値は、  
 認証方法 : Container  
 共用スコープ : Shareable
- 考慮点
  - ◆ WASランタイムでは、java:comp/env/jdbc/ResRef1と理解される
    - ▶ 参照名とJNDIのマッピングは、ibm-web-bnd.xmlファイルにて指定する
  - ◆ アプリケーション・コード内にて、認証方式と共用スコープが指定できる

© 2012 IBM Corporation

126

WASV8では、EJB3.0 / Servlet2.5をサポートしており、アノテーションを使用してJNDIをlookupすることができます。アノテーションとは、ソースコード中にコードの属性情報を付加する機能です。概念自体は以前からありましたが、J2SEの文法としてはJ2SE5.0から導入されました。また、アノテーションとして記述された属性情報は、コンパイル後もクラスファイル中に残り、それを解釈するツール(EJBであればコンパイラやEJBコンテナ)と組み合わせられ機能を発揮します。

EJB3.0の仕様書 一部抜粋

Definition of the Java language metadata annotations that can be used to annotate EJB applications. These metadata annotations are targeted at [simplifying the developer's task](#), at [reducing the number of program classes](#) and interfaces the developer is required to implement, and at [eliminating the need for the developer to provide an EJB deployment descriptor](#).

アノテーションを使用している場合のインフラ担当者の考慮点として、以下が挙げられます。

- ・アプリケーション・コード内に指定した認証方式と共用スコープの設定を確認する
- ・参照名とJNDIのマッピングは、ibm-web-bnd.xmlファイルにて指定する

また、@Resourceの属性は以下になります。

name	リソースの JNDI 名を指定する。
type	リソースの Java データ型を指定する。
authenticationType	このリソースに使用する認証タイプを指定する。
shareable	リソースを、共用できるかどうかを指定する。
mappedName	Bean参照をマッピングする。
description	リソースの説明を指定する。

Designing Web System Infrastructure with WAS V8.0

## 共用スコープ

- 共用可能接続 (デフォルト値)
  - ◆ 同じトランザクション・スコープ内で、`getConnection()`を呼び出すと、リソース参照の設定が同じ場合、毎回同じ接続を取得する
    - ▶ グローバル・トランザクション内で`close`しても、スコープを抜けるまで接続オブジェクトは接続プールに戻りません
    - ▶ 接続オブジェクトのプロパティ(例:分離レベル)を変更すると、`SharingViolationException`が発生する
- 共用不可能接続
  - ◆ 毎回異なる接続オブジェクトを取得する
    - ▶ グローバル・トランザクション内で接続オブジェクトを`close`すると、接続オブジェクトはグローバル・トランザクションの終了を待ちます
- 選択指針
  - ◆ パフォーマンス・テストにて接続プールが枯渇していないかを確認する
    - ▶ 共用不可能接続の方が、効率よく接続プールを使い回しできる可能性が高い
    - ▶ 共用スコープを変更する場合には、アプリケーションへの影響を考慮する

127

© 2012 IBM Corporation

共用可能接続では、同じトランザクション・スコープ内で同じ接続を取得できます。ただし、接続オブジェクトのプロパティ(`readOnly`属性、分離レベル等)を変更すると、`Sharing Violation`例外が発生してします。また、グローバル・トランザクション内で接続を`close`しても、トランザクション・スコープを抜けるまで接続プールには戻らず、上記スライド部分の図ではClassAが終了するまでその接続オブジェクトを他のアプリケーションが使用できませんので、ご注意ください。

(以下、InfoCenterより)

共用可能接続にした場合アプリケーションのなかには、共用可能接続を使用できるものがあります。それ以外のアプリケーションでは、特に、単一の`service()`メソッド内で複数の接続を取得するサーブレットでは、共用可能接続の使用は好ましくありません。

共用不可能接続にした場合は、毎回異なる接続オブジェクトを取得できます。物理接続と接続オブジェクトが1:1のため、アプリケーション・コード上もわかりやすくなります。接続オブジェクトは、アプリケーション・コード内でキャッシュする場合に有効活用出来ます。

・`Sharing Violation`例外発生時のエラー出力

```
[08/08/13 20:02:27:187 JST] 00000028 SystemErr    R Caused by:
javax.resource.spi.SharingViolationException: DSRA9250E: オペレーション setTransactionIsolation
は、Shareable Connections のグローバル・トランザクション中に許可されません。
```

Designing Web System Infrastructure with WAS V8.0

## 共用スコープの設定

- 設定方法
  - ◆ java:comp/env/によるJNDIのlookupの場合
    - デプロイメント記述子の共用スコープに設定する
      - 共用可能接続 Shareable
      - 共用不可能接続 UnShareable



```
<resource-ref>
  <res-ref-name>jdbc/ResRef1</res-ref-name>
  <res-auth>Container</res-auth>
  <res-sharing-scope>Unshareable</res-sharing-scope>
</resource-ref></web-app>
```

- ◆ アノテーションによるJNDIのlookupの場合
  - アプリケーション・コード内(アノテーションの属性)に設定する
    - 共用可能接続 Shareable=true
    - 共用不可能接続 Shareable=false

```
@Resource(name="jdbc/ResRef1", shareable=false)
```

© 2012 IBM Corporation 128

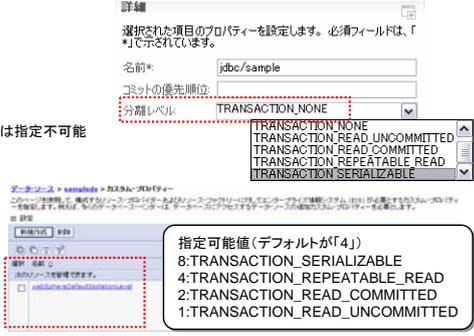
共用スコープは、java:comp/env/によるJNDIのlookupの場合はデプロイメント記述子に、アノテーションによるJNDIのlookupの場合にはアプリケーション・コード内に設定して下さい。

Designing Web System Infrastructure with WAS V8.0

## JavaEEアプリケーションにおける分離レベルの設定 (1/2)

- 設定方法
  - ◆ CMP Entity Beanの場合
    - アクセス・intent選択による自動指定 (→次ページ参照)
  - ◆ 上記以外のWebModuleやEJB (Session Bean、BMP)
    - リソース参照で指定
      - ① ibm-web-ext.xml
    - アプリケーション内で指定
      - ② setTransactionIsolation()
      - 共有可能接続で2度目以降の接続の場合は指定不可能
      - ③ SQL文のwith句
    - 管理コンソールで指定
      - ④ データ・ソースのカスタムプロパティ

・ANSI/ISO標準トランザクション分離レベル  
Uncommitted Read (未コミット読み取り)  
Read Committed (コミット読み取り)  
Repeatable Read (繰り返し可能読み取り)  
Serializable (直列可能)



指定可能値 (デフォルトが「4」)

8:TRANSACTION\_SERIALIZABLE  
4:TRANSACTION\_REPEATABLE\_READ  
2:TRANSACTION\_READ\_COMMITTED  
1:TRANSACTION\_READ\_UNCOMMITTED

- 設定指針
  - ◆ リソース参照で指定することを推奨
    - 優先順位は ③ > ② > ① > ④

© 2012 IBM Corporation
129

分離レベルとは、複数の処理が同時に実行された場合、他の処理からの影響度合いを定義したレベルになります。

Java EEアプリケーションでデータベースを使用する場合、分離レベルの指定はベンダー依存部分であり、WASでの指定方法はCMPとそれ以外で異なります。

CMP Entity Beanでは、アクセス・intentで指定します。WASV4.0までと異なりWASV5.0でのアクセス・intentは、Update意図の有無、Optimistic(WAS側で検索時点で更新前値を保管し、更新時に変更がされていないかチェックを行う機能)かPessimistic(DBでの排他制御を利用)かの定義、分離レベルなど、事前に定義されているwsOptimisticReadといったパラメーターの組み合わせを指定します。デフォルトは、wsPessimisticUpdate=WeakestLockAtLoadで、ここで選択される分離レベルは[REPEATABLE\_READ]です。アクセス・intentは、Beanのデフォルトおよびメソッド単位に指定できます。Entity Beanとアクセス・intentの詳細に関しては、WAS V5 Announcement Workshopの『Persistence Manager』や、InformationCenterをご確認下さい。

CMP以外のEJBモジュールであるSession BeanやBMP、ServletなどのWebモジュールでの分離レベルは、複数箇所を設定することができます。アプリケーション内部でsetTransactionIsolation()メソッドなどを使用して分離レベルを変更することも可能ですが、設定値の一元管理という観点からリソース参照で指定し、必要に応じてSQL文のwith句やsetTransactionIsolation()により設定変更する方法が推奨されています。複数箇所を設定した場合の優先順位は、SQL文のwith句 → setTransactionIsolation() → リソース参照 → 管理コンソール(データ・ソースのカスタム・プロパティ)です。

また、Oracle JDBC Driverでは、サポートするIsolation levelがREAD\_COMMITTEDとSERIALIZABLEの2つになってしまうため、デフォルトのREPEATABLE\_READはSERIALIZABLEとなります。このままでは通常のアプリケーションでは競合が起きやすく、パフォーマンスの低下が予想されるので必ず適切な値(多くのケースではREAD\_COMMITTED)を指定するようにして下さい。

Designing Web System Infrastructure with WAS V8.0

## JavaEEアプリケーションにおける分離レベルの設定 (2/2)

■ アクセス・インテント

アクセス・インテント・プロファイル	分離レベル						更新ロック
	DB2	Oracle	SyBase	Informix	Cloudscape	SQL Server	
wsPessimisticUpdate-WeakestLockAtLoad (デフォルト・ポリシー)	RR	RC	RR	RR	RR	RR	No ※OracleのみYes
wsPessimisticUpdate	RR	RC	RR	RR	RR	RR	Yes
wsPessimisticRead	RR	RC	RR	RR	RR	RR	No
wsOptimisticUpdate	RC	RC	RC	RC	RC	RC	No
wsOptimisticRead	RC	RC	RC	RC	RC	RC	No
wsPessimisticUpdate-NoCollisions	RC	RC	RC	RC	RC	RC	No
wsPessimisticUpdate-Exclusive	S	S	S	S	S	S	Yes

■ 考慮点

**注意**

- ◆ JDBCの分離レベルとDB2の分離レベルは異なる
- ◆ WASV5.0以降では、JDBCのデフォルト分離レベルは「TRANSACTION\_REPEATABLE\_READ」(DB2のRS)
  - ▷ DB2のデフォルト分離レベルはCS(CC)であるが、WAS側(アプリケーション)の設定が優先される

	JDBC		DB2
TRANSACTION_SERIALIZABLE	S	RR	Repeatable read
TRANSACTION_REPEATABLE_READ	RR	RS	Read stability
TRANSACTION_READ_COMMITTED	RC	CS	Cursor stability
TRANSACTION_READ_UNCOMMITTED	RU	UR	Uncommitted read

分離レベルの確認方法は、巻末の参考資料をご参照下さい。

© 2012 IBM Corporation 130

JDBCとDB2の分離レベルは異なりますのでご注意ください。(特に、JDBCとDB2の両方に同一の略語であるRRがあります。)

また、WASV5.0以降では、JDBCのデフォルト分離レベルがTRANSACTION\_REPEATABLE\_READ(DB2のRS)になり、DB2のデフォルト分離レベルであるCSと異なります。下記リンク先をご確認ください。(WASV4は、「TRANSACTION\_READ\_COMMITTED」(DB2のCS))

・WAS V5のデフォルトIsolation Levelの変更(CS → RS)による注意点 (DM-04-025)

<http://www.ibm.com/jp/domino01/mkt/cnpages1.nsf/page/default-003B6578?OpenDocument&ExpandSection=-1>

さらに、resultSetHoldabilityにつきましても、DB2やOracleのデフォルトは、resultSetHoldability=1 (HOLD\_CURSORS\_OVER\_COMMIT) ですが、WASがカスタム・プロパティresultSetHoldability を2(CLOSE\_CURSORS\_AT\_COMMIT)で上書きしますので、ご注意ください。

- HOLD\_CURSORS\_OVER\_COMMIT トランザクションをコミット後でもカーソルをオープンしたままにします。

- CLOSE\_CURSORS\_AT\_COMMIT トランザクションがコミットした時にカーソルをクローズします。

・WASV8.0 Information Center - 「Java Persistence API (JPA) アプリケーションのトラブルシューティング」

[http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tejb\\_jpatroubleshoot.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tejb_jpatroubleshoot.html)

Designing Web System Infrastructure with WAS V8.0

## 分離レベルの確認方法

- Statementのイベント・モニター (データベース)
 

```
Type : Dynamic
Operation: Execute
Section : 2
Creator : NULL ID
Package : SYSSN300
Consistency Token : SYSLVL01
Package Version ID :
Cursor : SQL_CURSN300C2
Cursor was blocking: FALSE
Text : UPDATE JPATEST.ACCOUNT SET amount = ? WHERE account_id = ?
```

DB2V8.1以降  
 パッケージ名「SYSXXNYY」のN部分を見る  
 0 = NC(コミットなし), 1 = UR, 2 = CS, 3 = RS, 4 = RR  
 この場合SYSSN300で「3」なのでISOLATIONはRS
- JDBCトレース (JDBCドライバー)
 

```
[ibm][db2][jcc][Time:xxx][Thread:WebContainer : 0][Connection@xxx]
[省略] getMetaData () returned DatabaseMetaData@7aa67aa6
[省略] isInDB2UnitOfWork () returned false
[省略] getHoldability () returned 2
[省略] getAutoCommit () returned true
[省略] getCatalog () returned null
[省略] isReadOnly () returned false
[省略] setTransactionIsolation (2) called
[省略] clearWarnings () called
[省略] setAutoCommit (false) called
```

setTransactionIsolationの()部分を見る  
 0 = NC(コミットなし), 1 = UR, 2 = CS, 4 = RS, 8 = RR  
 この場合「2」なのでISOLATIONはCS

© 2012 IBM Corporation 131

分離レベルの確認方法として、DB2側でStatementのイベント・モニターと、JDBCドライバー側でJDBCトレースがあります。Statementのイベント・モニターでは、SQL文のwith句で分離レベルを指定した場合には、その設定を確認することができませんのでご注意ください。

Designing Web System Infrastructure with WAS V8.0

---

## 2. WAS-DB2接続設計

- DB2 System Monitor
- WASトレース設定
- JDBCトレース設定

- ・WAS-DB2接続設計
  - JDBCドライバー設計
  - データソース設計
  - アプリケーション設計
  - パフォーマンス / 問題判別
  - Hints & Tips
- ・WAS-Oracle接続設計

© 2012 IBM Corporation 132

Designing Web System Infrastructure with WAS V8.0

### 性能監視に使えるツール ~WAS-DB2の処理時間を把握する

1. DB2 System Monitor
2. WASTレース設定
3. JDBCトレース設定

The diagram illustrates the data flow and monitoring points between WAS and DB2. A Client connects to a Web Server (containing a plugin) which connects to WAS (Web Container and Data Source). WAS connects to a JDBC Driver, which connects to a DB2 Client, which finally connects to a DB2 Server. Monitoring is shown with dashed boxes: WAST traces are located between the Web Server and WAS, while JDBC traces and DB2 System Monitor are located between the JDBC Driver and DB2 Client.

© 2012 IBM Corporation 133

WAS-DB2間の処理が遅いというときに原因となっている箇所を特定するために利用できるトレース機能についてご紹介します。

■DB2 System Monitor

各コンポーネントでの処理時間を取得できる

■WASTレースの取得方法

どのSQLやEJB呼び出しが実行されているかを確認できる

■JDBCトレースの取得方法

JDBCドライバーとDBサーバー間の通信にかかっている時間を見る、JDBCドライバー上の処理の流れを確認する

詳細は、巻末資料をご参照ください。

Designing Web System Infrastructure with WAS V8.0

## DB2 System Monitor

- JDBCアプリケーション、もしくはWAS管理コンソールからデータベース・システム・モニタリングを設定することができる機能
  - ◆ アプリケーションの経過時間
  - ◆ JDBCドライバーの経過時間
  - ◆ ネットワークI/Oの経過時間
  - ◆ DB2サーバーの経過時間
- 設定指針
  - ◆ パフォーマンス低下のボトルネック箇所を特定したい場合
- 設定方法
  - ◆ 1. アプリケーションで実装する
  - ◆ 2. JDBCトレースを取得する

App time    Driver time    I/O time    Server time

134

© 2012 IBM Corporation

System Monitorとは、JDBCアプリケーション、もしくは管理コンソールからデータベース・システムをモニターすることができる機能です。DBアクセス処理にて、パフォーマンス低下のボトルネック箇所を特定したい場合に当機能を利用することをご検討下さい。

・DB2V9.7 InformationCenter - 「DB2SystemMonitor インターフェース」

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.apdv.ia.va.doc/doc/r0021838.html>

Designing Web System Infrastructure with WAS V8.0

## DB2 System Monitor 取得 / 設定方法

- アプリケーション実装
 

```
DB2SystemMonitor monitor=((DB2Connection)conn).getDB2SystemMonitor();
monitor.enable(true);
monitor.start(com.ibm.db2.jcc.DB2SystemMonitor.RESET_TIMES);
```

⇒ ここに実行するSQLを記述する

```
monitor.stop();
monitor.getServerTime()
monitor.getNetworkIOTime()
monitor.getCoreDriverTime()
monitor.getApplicationTime()
```

<出力例>  
 Server elapsed time (microseconds)=560  
 Network I/O elapsed time (microseconds)=3633  
 Core driver elapsed time (microseconds)=4331  
 Application elapsed time (milliseconds)=4
- JDBCトレース
  - ◆ 管理コンソールより、JDBC -> データ・ソース -> カスタム・プロパティを選択し、以下を設定する (→後述)
    - > traceLevel     -1 or 131072
    - > tracefile       /work/jdbc\_trace.log (任意)

<出力例>  
 [ibm][db2][jcc][SystemMonitor:start]  
 [ibm][db2][jcc][Time:2009-02-13-15:58:02.852][Thread:WebContainer : 0][Statement@79007900]  
 execute (SELECT \* FROM STAFF) called  
 [ibm][db2][jcc][SystemMonitor:stop] core: 70.639ms | network: 52.346999ms | server: 0.0ms

© 2012 IBM 135

アプリケーションの実装方法は、スライド部分を参考にして下さい。出力例部分は、下記のようにコーディングしています。

```
System.out.println("Server elapsed time (microseconds)=" + systemMonitor.getServerTimeMicros());
System.out.println("Network I/O elapsed time (microseconds)=" +
systemMonitor.getNetworkIOTimeMicros());
System.out.println("Core driver elapsed time (microseconds)=" +
systemMonitor.getCoreDriverTimeMicros());
System.out.println("Application elapsed time (milliseconds)=" +
systemMonitor.getApplicationTimeMillis());
```

JDBCトレースは、管理コンソールより、JDBC -> データ・ソース -> カスタム・プロパティを選択し、traceLevelとtraceFileを設定して下さい。(JDBCトレースについては、後述致します。) 出力例の「Server」については、DB2® Database for Linux, UNIX, and Windows バージョン 9.5 以降、およびDB2 for z/OSにてサポートされます。

Designing Web System Infrastructure with WAS V8.0

## WASTレース設定

	取得できる項目	設定箇所/方法	使用例
①	データベースの接続状況を確認する ・実行されたSQL文 ・トランザクションの開始、コミット・ロールバック ・EJB 呼び出し (Create、Remove、findByPrimaryKeyなど)	<WASTレース設定 / ログ詳細レベルの変更> *= <code>info:WAS.clientinfopluslogging=all</code>	パフォーマンス・テストなど、実行されているSQLをWAS上で確認したい場合には、ログ出力量が少なく便利である。ClientInformationAPIの設定情報も確認できる。
②	データベースの接続状況を確認する ・①の取得項目 ・接続プールの使用状況 ・SQL文のレスポンス・タイム ・リソース参照の情報 ・トランザクションの開始・終了	<WASTレース設定 / ログ詳細レベルの変更> *= <code>info:WAS.j2c=all:RRA=all:WAS.database=all:Transaction=all</code>	①よりも詳細な情報が必要な場合に設定する
③	データベースの接続状況、およびJDBCドライバの状況を確認する ・②の取得項目 ・DB2側のApplicationID ・パラメーター・マーカの設定値 ・ネットワーク、Driver内の処理時間 (DB2V9.5以降)	<WASTレース設定 / ログ詳細レベルの変更> *= <code>info:WAS.j2c=all:RRA=all:WAS.database=all:Transaction=all</code> <JDBCレース設定> <code>traceLevel = -1</code> <code>traceFile</code> は設定しない	②よりも詳細な情報や、JDBCドライバーに関する情報が必要な場合に設定する

© 2012 IBM Corporation 136

WASで取得できるデータベースに関連するトレース設定をまとめます。上記スライド部分を参考にし、状況に応じたトレース設定を行って下さい。

・MustGather: Read first for all WebSphere Application Server products

<http://www.ibm.com/support/docview.wss?rs=180&uid=swg21145599>

Designing Web System Infrastructure with WAS V8.0

## WASトレース取得例

**トレース①**

```
[09/02/25 10:36:43:984 JST] 00000038 clientinfoplus > prepareStatement Entry
com.ibm.ws.rsadapter.jdbc.WSJccSQLJConnection@43124312
SELECT count(*) FROM syscat.tables,syscat.tables,syscat.tables
TYPE FORWARD ONLY (1003)
CONCUR_READ_ONLY (1007)
[09/02/25 10:36:44:046 JST] 00000038 clientinfoplus < prepareStatement Exit
com.ibm.ws.rsadapter.jdbc.WSJccPreparedStatement@39ca39ca
[09/02/25 10:36:44:046 JST] 00000038 clientinfoplus 3 .setClientInformation(Properties props,WSRdbManagedConnectionImpl mc, boolean explicitCall) with
sqlConn {CLIENT_APPLICATION_NAME=tmp_appl, CLIENT_ID=WASUSR1}
```

•SQL Strings  
•ClientInformation API

**トレース②**

```
[09/02/24 14:55:38:484 JST] 00000038 PoolManager 3 reserve()
PoolManager name:jdbc/testDS
PoolManager object:1590132124
Total number of connections: 2 (max/min 10/1, reap/unused/aged 180/1800/0, connectiontimeout/purge 180/EntirePool)
(testConnection/interval false/0, stuck timer/time/threshold 0/0/0, surge time/connections 0/-1)
Shared Connection information (shared partitions 200)
com.ibm.ws.LocalTransaction.LocalTranCoordImpl@71957d83:RUNNING: MCWrapper id 20fb3d86 Managed connection
WSRdbManagedConnectionImpl@2ec3fd81 State:STATE_ACTIVE_INUSE
Total number of connection in shared pool: 1
Free Connection information (free distribution table/partitions 5/1)
(2)(0)MCWrapper id 7828fd84 Managed connection WSRdbManagedConnectionImpl@750bfd86 State:STATE_ACTIVE_FREE
Total number of connection in free pool: 1
```

•接続プールの使用状況

**トレース③**

```
[09/02/23 20:20:37:703 JST] 0000002e logwriter 3 [ibm][db2][jcc][Time:1816812037703][Thread:WebContainer : 0][ResultSet@60866086] next () called
[09/02/23 20:20:37:718 JST] 0000002e logwriter 3 [ibm][db2][jcc][Time:1816812037718][Thread:WebContainer : 0][ResultSet@60866086] next () returned true
[09/02/23 20:20:37:734 JST] 0000002e logwriter 3 [ibm][db2][jcc][Time:1816812037734][Thread:WebContainer : 0][ResultSet@60866086] getInt (ID) called
(省略)
{jcc}[Connection@8fc08fc] DB2 LUWID: 9.188.198.118.56189.08042109292.0001
(省略)
[ibm][db2][jcc][SystemMonitor:stop] core: 94.63ms | network: 48.102ms | server: 40.508ms
```

•JDBCトレース  
•DB2側のApplicationID  
•Driver内の処理時間

© 2012 IBM Corporation 137

本編p48のWASトレース設定毎の出力例となります。

<WASトレース設定 / ログ詳細レベルの変更>

- トレース①      \*=info:WAS.clientinfopluslogging=all
- トレース②      \*=info:WAS.j2c=all:RRA=all:WAS.database=all:Transaction=all
- トレース③      \*=info:WAS.j2c=all:RRA=all:WAS.database=all:Transaction=all

<JDBCトレース設定>

- トレース③      traceLevel=-1

Designing Web System Infrastructure with WAS V8.0

## JDBCトレース設定 (DB2 JDBC Driver)

- JDBCドライバーとデータベース間でやり取りされるデータの流れを取得
  - ◆ ドライバーのプロパティ値、実行しているSQLステートメント、実行しているJDBC API等を確認できる
- 設定指針
  - ◆ 以下のようなWASトレース設定では取得できない場合
    - ドライバーの内部エラー
    - JDBC APIの実行時刻とreturn時刻
    - パラメーター・マーカの設定値
- 設定方法
  - ◆ 管理コンソールより、JDBC → データ・ソース → カスタム・プロパティを選択し、以下を設定する

```
[ibm][db2][cc][省略][PreparedStatement@39eae493] setInt(1,1) called
[ibm][db2][cc][省略][PreparedStatement@39eae493] setString(2,COL2*1) called
```

<input type="checkbox"/> traceLevel	Specifies the level of trace, determined by a bitwise combination of constants: TRACE_NONE=0, TRACE_CONNECTION_CALLS=1, TRACE_STATEMENT_CALLS=2, TRACE_RESULT_SET_CALLS=4, TRACE_DRIVER_CONFIGURATION=16, TRACE_CONNECTIONS=32, TRACE_SQL_ROWS=64, TRACE_RESULT_SET_META_DATA=128, TRACE_PARAMETER_META_DATA=256, TRACE_DIAGNOSTICS=512, TRACE_SQL=1024, TRACE_META_CALLS=192, TRACE_DATASOURCE_CALLS=16384, TRACE_LARGE_OBJECT_CALLS=13728, TRACE_SYSTEM_MONITOR=13727, TRACE_TRACEPOINTS=102144, TRACE_ALL=1.	false
<input type="checkbox"/> traceFileAppend	Specifies whether to append to or overwrite the file specified by the traceFile property. The default is false, which means the file specified by the traceFile property is overwritten.	false
<input type="checkbox"/> traceFile	The file to store the trace output. If you specify the trace file, the DB2 JDBC trace will be logged in the file. If the property is not specified and the WAS database or com.ibm.was.db2.tracer.trace group is enabled, then both WebSphere Application Server trace and DB2 trace will be logged into the WebSphere Application Server trace file.	false
<input type="checkbox"/> traceDirectory	Specifies the directory where the trace file will be created.	false

トレースレベルを設定する。  
整数値を加算することで、複数のトレースが取得可能。  
(例)1+2=3 (CONNECTION\_CALL, STATEMENT\_CALL)

trueに設定すると、トレース・ファイルが上書きされない。

トレースファイルを設定する。

接続オブジェクト毎にトレース・ファイルが出力される。  
(例)\_cpds\_1, \_cpds\_2...  
traceFileを設定すると、traceDirectoryは無効になる。

138

JDBCトレースとは、JDBCドライバーとデータベース間でやり取りされるデータの流れを取得したものになります。ドライバーの内部エラー、JDBC APIの実行時刻とreturn時刻やパラメーター・マーカの設定値を確認したい場合等に、取得することを検討して下さい。

JDBCトレースは、管理コンソールより、JDBC → データ・ソース → カスタム・プロパティを選択し、traceLevel、traceFileAppend、traceFile、traceDirectoryを設定して下さい。traceLevelについては、下記リンク先をご確認下さい。

・DB2 Information Center - 「サポートされるすべてのデータベース製品に共通の IBM Data Server Driver for JDBC and SQLJ のプロパティ」

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.apdv.ia.va.doc/doc/r0052038.html>

(※)パラメーター・マーカとは

通常、疑問符(?)で示され、ステートメント実行中に値が取得されるSQLステートメント内のプレース・ホルダーです。何度も実行する必要のあるSQLステートメントの場合、SQLステートメントを一回だけ準備し、パラメーター・マーカを使って実行時に入力値を置換することにより、照会プランを再利用することが出来ます。

○ SELECT \* FROM T1 WHERE C1 = ? ;

× SELECT \* FROM T1 WHERE C1 = 100;

・DB2V9.7 InformationCenter - 「パラメーター・マーカ」

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.apdv.ro.utines.doc/doc/c0020295.html>

## まとめ:パフォーマンス / 問題判別

項目	選択肢	設計指針
DB2 System Monitor	<ul style="list-style-type: none"> <li>・アプリケーション実装</li> <li>・JDBCトレース</li> </ul>	パフォーマンス低下のボトルネック箇所を特定したい場合に、設定することを検討して下さい。
WASTレース	<ul style="list-style-type: none"> <li>①データベースの接続状況を確認する</li> <li>②(①よりも詳細な)データベースの接続状況を確認する</li> <li>③データベースの接続状況、およびJDBCドライバーの状況を確認する</li> </ul>	<p>パフォーマンス・テストなど、実行されているSQLをWAS上で確認したい場合に①を設定することを検討して下さい。</p> <p>①よりも詳細な情報が必要な場合に②を設定することを検討して下さい。問題判別を行う際に使用して下さい。</p> <p>②よりも詳細な情報や、JDBCドライバーに関する情報が必要な場合に③を設定することを検討して下さい。②と同様に、問題判別を行う際に使用して下さい。</p>
JDBCトレース	<p>データ・ソースのカスタム・プロパティ</p> <ul style="list-style-type: none"> <li>・traceLevel など</li> </ul>	<p>以下のような、WASTレース設定では取得できない情報を取得したい場合に設定することを検討して下さい。</p> <ul style="list-style-type: none"> <li>・ドライバーの内部エラー</li> <li>・JDBC APIの実行時刻とreturn時刻</li> <li>・パラメーター・マーカの設定値</li> </ul>

- WAS-DB2接続のタイムアウト設定
- ClientInformationAPIによるワークロード管理
- DB2 64bitインスタンスへの接続方法
- 監査ログ取得例

## 2. WAS-DB2接続設計

### ・WAS-DB2接続設計

- JDBCドライバー設計
- データソース設計
- アプリケーション設計
- パフォーマンス / 問題判別

### - Hints & Tips

- ・WAS-Oracle接続設計

Designing Web System Infrastructure with WAS V8.0

## WAS-DB2接続タイムアウトに関わる設定

- 「タイムアウト」= 処理時間に一定の上限を設けておくこと
  - ◆ ある処理に時間がかかり過ぎた場合に、処理を中断しエラーを返すことで、ユーザーに処理の失敗を通知する
  - ◆ タイムアウトがあることで、クライアントがサーバーからの応答を無限に待たずに済む ⇒ 状況がわかる・時間がたったらリトライするなどのアクションが取れる
- お客様要件にあわせて適切な設定値をご検討ください。
  - ◆ タイムアウトが長すぎると、クライアント(ブラウザ)への応答が長時間戻らない
  - ◆ タイムアウトが短すぎると、必要な処理が常に中断されエラーとなる

© 2012 IBM Corporation 141

タイムアウトは、サーバーの高負荷状態やネットワーク障害時などに、クライアントが無限に待つ事態を発生させないために必要な考え方です。

WASからDB2への接続中におけるタイムアウト時間を決めるための設定項目として、当ページの図に記したようなパラメーターがあります。

Designing Web System Infrastructure with WAS V8.0

■□□ SQLの処理時間が想定範囲を超過している状況への対応 □□■

### JDBC標準: Statement#setQueryTimeoutメソッド

- Statement#setQueryTimeoutメソッド
  - ◆ JDBC APIにて実装される Statement.setQueryTimeout(int) メソッドにて想定処理時間(秒単位)を設定しておく、設定時間を超過してもQueryが終了しない時に該当のQueryをキャンセルし、アプリケーションにExceptionが返るよう制御できる
  - ◆ 接続先がDB2の場合、データ・ソースへのカスタム・プロパティによりQueryタイムアウト後の動作内容を設定することができる(次の2ページを参照)
- 考慮点
  - ◆ DBサーバー側のプロセスダウンやNW障害などに起因する障害等、ネットワーク接続断の状況には対応できない

© 2012 IBM Corporation

142

WAS V8では、executeQuery/executeUpdate発行後に実行完了を待つタイムアウト設定パラメータは提供されていません。

実際の業務のSQLが実行されてからの処理時間の上限を設けたい場合、JDBC標準の方法としてはStatement#setQueryTimeout()メソッドを用いて、該当クエリの実行時間に上限を設けることができるようになっています。

指定した時間を超過すると、クライアントには例外が戻されるため、クライアント側ではその後リトライするか否か、等の対応を検討することができます。

ただし、高負荷に起因するタイムアウトの場合には、Query実行をキャンセルさせるためのスレッドが即時に動作できず、タイムアウトが発生するまでに時間がかかる可能性があるため100%確実な方法ではないことをご留意ください。

なお、接続先がDB2で、DB2 Data Server Driver for JDBC and SQLJをお使いの場合には、データ・ソース・カスタム・プロパティとして、Queryタイムアウト時間を迎えた後の動作について設定を行うことができます。

Designing Web System Infrastructure with WAS V8.0

■□□ SQLの処理時間が想定範囲を超過している状況への対応 □□■  
**DB2 JDBCドライバー固有のパラメーター①**

以下のパラメーターは、Statement#setQueryTimeout(int)にて、SQLの実行時間がTimeout値を超過した後の挙動を決めるDB2 JDBCドライバー固有のパラメーターです。

- queryTimeoutProcessingMode (DB2 9.7 FP4 ~)
  - ◆ Statement#setQueryTimeoutで指定した時間が経過したときに、JDBCドライバーがSQLステートメントを取り消すか、接続をクローズするかを指定
  - ◆ **DB2BaseDataSource.INTERRUPT\_PROCESSING\_MODE STATEMENT\_CANCEL (1): デフォルト**
    - ▶ タイムアウト到達時点で実行されているSQLステートメントの実行をキャンセル
  - ◆ **DB2BaseDataSource.INTERRUPT\_PROCESSING\_MODE CLOSE\_SOCKET (2)**
    - ▶ タイムアウト到達時点でソケットを削除し、DB接続をクローズ

The diagram illustrates the flow of data and control between the application and the database. An Application on the left interacts with a Managed Connection, which is managed by a DataSource. The DataSource contains a Connection Pool and a JDBC Driver. The JDBC Driver connects to a Physical Connection on the DB Server. A DataStoreHelper is also shown. A red 'X' indicates that the Physical Connection is being closed. Labels 'setQueryTimeout' and 'queryTimeoutProcessingMode' point to the respective parameters in the diagram.

© 2012 IBM Corporation 143

DB2 JDBC Driver を利用されている場合には、Statement#setQueryTimeoutメソッドにてSQLの実行時間がタイムアウト値を超過したと判断された後の挙動をデータソースのプロパティとして定義することができます。

こちら設定値は1「キャンセル」か2「クローズ・ソケット」の2択となり、デフォルト設定は「1」でありタイムアウト到達時点でSQLステートメントの実行をキャンセルします。

「クローズ・ソケット」を選択すると、タイムアウト時点でソケットが削除されDB接続がクローズされます。

■ 参考リンク

DB2 9.7 新機能

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.wn.doc/doc/c0051316.html>

Designing Web System Infrastructure with WAS V8.0

■ □ □ SQLの処理時間が想定範囲を超過している状況への対応 □ □ ■  
DB2 JDBCドライバー固有のパラメーター②

以下のパラメーターは、Statement#setQueryTimeout(int)にて、SQLの実行時間がTimeout値を超過した後の挙動を決めるDB2 JDBCドライバー固有のパラメーターです。

■ interruptProcessingMode (Type4のみ)

- ◆ Statement#cancelが実行された際のJDBCドライバーの挙動を指定
- ◆ 設定値は下記3つ(詳細はノートを参照)
  - DB2BaseDataSource.INTERRUPT\_PROCESSING\_MODE\_DISABLED (0)
  - DB2BaseDataSource.INTERRUPT\_PROCESSING\_MODE\_STATEMENT\_CANCEL (1): デフォルト
  - DB2BaseDataSource.INTERRUPT\_PROCESSING\_MODE\_CLOSE\_SOCKET (2)

© 2012 IBM Corporation 144

setQueryTimeoutで設定したタイムアウト時間が経過した後は、キャンセル処理が実行されますが、interruptProcessingModeパラメーターでは、このときのキャンセル処理の内容を定義することができます。

DB2BaseDataSource.INTERRUPT\_PROCESSING\_MODE\_DISABLED (0) :

Statement#cancel が実行されても、JDBCドライバーは何もしない

DB2BaseDataSource.INTERRUPT\_PROCESSING\_MODE\_STATEMENT\_CANCEL (1) :

Statement#cancel が呼び出されると、処理中のSQLステートメントの実行をキャンセルする(デフォルト)

DB2BaseDataSource.INTERRUPT\_PROCESSING\_MODE\_CLOSE\_SOCKET (2) :

Statement#cancel が呼び出されると、JDBCドライバーは下記いずれかの動作を示す

- Client Reroute や Client Affinityが設定されていない環境の場合、  
JDBCドライバーはソケット削除と接続のCloseを行い、SQLExceptionをクライアントに戻すことでクライアントにDBサーバーとの接続が切断されたことを知らせる
- Client Reroute や Client Affinity が有効になっている環境では、JDBCドライバーはソケット削除と接続のCloseを行った後、再接続を試みる。再接続に成功したらSQLExceptionを返すことで、再接続に成功したことをクライアントに知らせる。  
このとき、JDBCドライバーはSQLステートメントの再実行は行わない。  
(enableSeamlessFailoverプロパティがDB2BaseDataSource.YES に設定されていても再実行されない)

Designing Web System Infrastructure with WAS V8.0

■□□ データベース・サーバー、ネットワーク障害等に起因する接続断への対応 □□■  
**DB2 JDBCドライバー固有のパラメーター③**

- **loginTimeout (Type4のみ)**
  - ◆ データベースへの接続が確立できるまでに待つ秒単位の時間
  - ◆ 設定時間が経過すると、JDBCドライバーが接続をクローズ
  - ◆ デフォルトは0秒で、タイムアウト値はデフォルトのシステム・タイムアウト値となる
- **blockingReadConnectionTimeout (Type4のみ)**
  - ◆ 接続ソケットの読み取りがタイムアウトになるまでの秒単位の時間
  - ◆ デフォルトは0秒で、タイムアウトしない設定
  - ◆ アプリケーションで最長の照会を実行するのに必要な時間より数秒大きい値とする  
⇒ DB接続用ソケットにおける読み込み処理が開始されるまでに待つ時間とする
  - ◆ 注意点:長時間かかっているSQL処理(接続障害ではないもの)もタイムアウトとなる

© 2012 IBM Corporation 145

こちらDB2 JDBCドライバー固有のパラメーターとなりますが、初期接続確立時にDBサーバーからの応答を待つパラメーター loginTimeout と、Query実行時のタイムアウトを定義できるblockingReadConnectTimeoutとが提供されています。blockingReadConnect Timeoutの利用上の考慮点としては、データ・ソース単位の設定となるため、元々長く時間がかかるSQLについても強制的にタイムアウト扱いとなりクライアントにエラーが返ります。

■ 参考リンク

DB2 9.7 Javaアプリケーション開発ガイド

[ftp://ftp.software.ibm.com/ps/products/db2/info/vr97/pdf/ja\\_JP/DB2Dev.Java-db2ajj970.pdf](ftp://ftp.software.ibm.com/ps/products/db2/info/vr97/pdf/ja_JP/DB2Dev.Java-db2ajj970.pdf)

Designing Web System Infrastructure with WAS V8.0

■□□ データベース・サーバー、ネットワーク障害等に起因する接続断への対応 □□■  
**OS KeepAlive関連パラメーターのチューニングについて(1)**

- WASノードに適切なOSのKeepAliveパラメーターのチューニングが行われていない場合に発生しうる問題とは？
  - ◆ データベース障害時(※)に、WASのスレッドが長時間実行状態(応答待ち状態)で残存し、パフォーマンスの悪化やWASのサービス提供不能状態に繋がる可能性がある

(※): データベース障害時; ページング発生によるデータベース無応答、ノード障害時、ネットワーク障害時など

- WASスレッドを解放するためには、WASノードのOSのネットワーク・パラメーターのチューニングにより、非活動状態にある接続を強制的に切断し、アプリケーション・サーバーに処理を戻すことが必要

© 2012 IBM Corporation 146

先ほどのページまででご紹介したsetQueryTimeoutやblockingReadConnectionTimeoutなどによる明示的なタイムアウトが設定されていない場合、WAS-DBサーバ間のネットワーク障害やDBサーバ障害時など、WASから見てDBサーバから応答が返らない状況では、WASはTCP Keepalive Timeoutに到達するまでDBサーバからの応答を待ち続けます。

特にTCP/IPのタイムアウトはデフォルトで長い値が採用されているケースが多く(例: Linuxでは7200秒)となっており、DBサーバ障害がおきた際には長時間クライアントに回答が戻らない可能性があります。

(⇒次ページへ続きます)

Designing Web System Infrastructure with WAS V8.0

■□□ データベース・サーバー、ネットワーク障害等に起因する接続断への対応 □□■  
**OS KeepAlive関連パラメーターのチューニングについて(2)**

- 多くのOSで、デフォルトのTCP KEEPALIVEタイムアウトは非常に長い  
 ため、環境に応じてチューニングを実施してください。
  - ◆ 不必要な切断を防ぐため、ネットワーク機器のテークオーバー時間より長く  
 設定する
  - ◆ テークオーバー完了時に新規処理を受けられるよう、データベースのテイク  
 オーバー時間より短く設定する
  - ◆ OS全体に効くパラメーターであり、TCP/IPを利用するアプリケーションは全  
 て影響を受ける可能性があるため、注意する
- AIXの場合のデフォルト値とWASの挙動など、詳細はTechFlashを参照  
 (WAS-09-048)

© 2012 IBM Corporation 147

DBサーバへの接続断絶により長時間待ち時間が発生しないための設定・考慮点として、OSのKeepAliveタイムアウト関連のパラメータを短くしていただくことが一つの選択肢となります。ただし、そのOS上で動作するすべてのアプリケーションが影響を受けますので、パラメータ変更(短くする)を

実施いただく場合には、影響範囲について十分ご調査いただくこと、さらに負荷試験、長稼動試験を行った上問題がないことをご確認いただけますことをおすすめいたします。

・テクニカル・フラッシュ

「【注意事項】DB障害時におけるKeepAliveパラメータ設定のWASサービスへの影響について (WAS-09-048)」

<https://www-304.ibm.com/support/docview.wss?uid=jpn1J1004347>

Designing Web System Infrastructure with WAS V8.0

## ■□□ DBサーバのリソース消費を抑えるための実行制御のしくみ □□■ DB2 Workload Manager

- DB2 Workload Managerとは
  - ◆ ワークロードを管理し、限られたリソースの有効活用とシステムの安定稼働を実現する機能
  - ◆ アプリケーションの種類、接続ユーザー、稼働タイプなどによってきめ細かくアクティビティを分類し、優先度の調整を行うことができる
  - ◆ しきい値を設け、システム・リソースを圧迫する異常なクエリーの停止や、キューイングなどによる実行制御ができる
- ACTIVITYTOTALTIME しきい値
  - ◆ 実行時間が設定値を越えるSQLを停止する

THRESHOLD th\_totalacttime  
WHEN ACTIVITYTOTALTIME > 5 MINUTES

© 2012 IBM Corporation 148

データベース・サーバー側で、SQL実行時間の閾値などを決めておくこともできます。

Workload Managerは、DBサーバが制御不能な状態にならないよう、事前に定義した項目/ルールに基づいてサーバー・リソースの消費が過剰とならないよう制限を設ける仕組みです。

例えばSQLの実行時間があらかじめ設定しておいた閾値を超えた場合に該当のSQLを停止するなど、業務特性に応じたルールを設けておくことで、

DBサーバBusy時のSQL処理の待ち時間が長期化して、クライアントに応答を戻せなくなる事態を防ぎます。

Designing Web System Infrastructure with WAS V8.0

### (参考) DB2 Workload Manager

- ワークロードを管理し、限られたリソースの有効活用とシステムの安定稼働を実現する。
  - 目的に応じてワークロードを分類
  - 実行優先度の制御
  - しきい値管理
  - 個別レポーティング

The diagram illustrates the DB2 Workload Manager (WLM) architecture. It shows two main execution environments: a 'High Priority Execution Environment' (top, yellow) and a 'Low Priority Execution Environment' (bottom, blue). Workloads are classified into 'High Priority Workload' (yellow circles) and 'Low Priority Workload' (blue circles). The high priority environment includes 'Execution Priority Control' and 'High Priority Execution Environment'. The low priority environment includes 'Execution Priority Control' and 'Low Priority Execution Environment'. A 'Simultaneous Execution Control' (同時実行数の管理) mechanism is shown between the environments. A 'Threshold Management' (しきい値管理) section notes that various types of thresholds (large-scale queries, runaway query suppression, simultaneous execution control, etc.) are managed. A 'Reporting' (レポーティング) section shows 'Reporting by Purpose' (目的に応じたレポーティング). A 'Monitoring' (監視) section shows 'Monitoring of Thresholds in Running Applications' (実行中アプリケーションのしきい値監視). A warning icon indicates that 'System-wide impact may occur, so large-scale processing should be stopped in advance' (システム全体に影響を与えてしまうような大規模処理は事前に実行停止).

© 2012 IBM Corporation

149

Designing Web System Infrastructure with WAS V8.0

## ClientInformationAPI

- Webアプリケーションにクライアント情報を設定し、その情報をデータベースに受け渡すことができる機能 (WASV6以降)
- 設定方法
 

```
import com.ibm.websphere.rsadapter.WSConnection;
(省略)
con = (WSConnection) ds.getConnection();
Properties props = new Properties();
props.setProperty(WSConnection.CLIENT_ID, userid);
props.setProperty(WSConnection.CLIENT_LOCATION, location);
props.setProperty(WSConnection.CLIENT_ACCOUNTING_INFO, accounting);
props.setProperty(WSConnection.CLIENT_APPLICATION_NAME, appname);
props.setProperty(WSConnection.CLIENT_OTHER_INFO, other_info);
props.setProperty(WSConnection.OTHER_CLIENT_TYPE, client_type);
con.setClientInformation(props);
```

WSConnectionをimportする

ここにクライアント情報を設定する
- 適用例
  - ◆ 監査ログ
    - WASとDB2の監査ログを一致することができる
  - ◆ ワークロード管理
    - Webシステム全体でのワークロード管理を実現できる

© 2012 IBM Corporation 150

\* 当ページは、セキュリティーの章でご紹介したClientInformation API (P26)と同一内容です。\*

ClientInformationAPIとは、Webアプリケーションにクライアント情報を設定し、その情報をデータベースに受け渡すことができる機能です。WASV6以降でサポートされます。

Webアプリケーション・コード内に、WSConnectionをimportし、setPropertyメソッドにてクライアント情報を明示的に設定して下さい。

この機能の適用例としましては、監査ログやワークロード管理が考えられ、詳細は次ページ以降で説明致します。

・WASV8.0 Information Center - 「setClientInformation(Properties) API によるクライアント情報の設定」

[http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tdat\\_clientinfotask.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tdat_clientinfotask.html)

・DB2V9.7 InformationCenter - 「WLM\_SET\_CLIENT\_INFO ストアードプロシージャ」

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.sql.rtn.doc/doc/r0053116.html>

## ClientInformationAPI適用例【ワークロード管理】

**DB2**

J2C認証データ

別名	ibmkenCellManager@1.1test1
ユーザー ID	db2inst1
パスワード	*****

・DB2 ClientInformationAPIとDB2ワークロード管理を紐付ける

・Webアプリケーション ClientInformationAPIにてクライアント情報を設定

データ・ソース

CLIENTID: Maryさん

CLIENTID: Jackさん

クライアント: Mary, Jack

**Webアプリケーション設定例**

```
import com.ibm.websphere.rsadapter.WSConnection;
con = (WSConnection) ds.getConnection();
Properties props = new Properties();
props.setProperty(WSConnection.CLIENT_ID, Jack);
con.setClientInformation(props);
```

**DB2ワークロード管理**

db2inst1さん	検索処理	優先度中
db2inst1さん	検索処理	優先度中

**DB2ワークロード管理 + ClientInformationAPI**

Maryさん	検索処理	優先度高
Jackさん	更新処理	優先度低

■ 優先制御例

- ◆ オンライン処理とバッチ処理
- ◆ 一般社員の処理とマネージャーの処理

**DB2ワークロード管理例**

- 例1) 見積もりコストが10万を超える場合は実行しない
- 例2) 並列度が10を超えた場合はキューに待機させる
- 例3) 異常に長い時間がかかっている処理を停止する
- 例4) 検索結果行が膨大な処理を検出し、停止する
- 例5) リクエスト毎にCPU使用率を制限する

© 2012 IBM Corporation 151

クライアント、WAS、DB2を使用した3層Webシステムでは、例えば、少数ユーザーの大量検索処理が実行されると、DBサーバーのリソースが占有され、多くのユーザー処理が阻害されサービス全面ダウンにまでつながってしまうケースがあります。この問題を解決するためには、ユーザーやアプリケーション等の条件により、DBサーバーのマシン・リソースの使用を制限させる方法が考えられます。しかしながら、前項と同様の理由により、DBサーバーへアクセスするユーザーはデータ・ソースに定義された代表ユーザー(db2inst1)となり、全て同一条件のアクセスとなるため制限されることが出来ません。DB2ガバナナーの機能によって、実行経過時間によりそのアクティビティーを強制終了させる方法も考えられますが、システム要件として強制終了できない場合やシステムの負荷状態により実行経過時間が大きく異なるといった問題があります。

この問題に対してClientInformationAPIを使用すると、クライアント・リクエスト毎にDB2のワークロード管理機能と紐付けることができるようになります。さらに、DB2のワークロード管理機能は、AIXのCPU優先度やI/Oプリフェッチ優先度に紐付けることができます。

- ・DB2のしきい値
  - 対象オブジェクトに対して、制限やアクションを定義する
  - Predictive (予測的しきい値)の評価: Estimated cost
  - Proactive (並列度のしきい値)の評価: Current database activities等
  - Reactive (反応的しきい値)の評価: Elapsed time, Rows returned, Temporary space等
- ・しきい値を超過した場合のアクション
  - 処理を継続、処理を停止から選択する
- ・優先度制御
  - CPU優先度 (Agent Priority)
  - I/Oプリフェッチ優先度 (Prefetch Priority)

Designing Web System Infrastructure with WAS V8.0

## DB2 64bitインスタンスへの接続方法

- DB2 64bitインスタンスを利用するメリット
  - ◆ 巨大なメモリー空間を利用可能
    - 大きなバッファ・プールが作成可能になり、処理が高速になる
- 32bitWAS - 64bitDB2の接続方法
  - ◆ ①Type 4ドライバー
    - TCP/IP経由で接続するため、特別に設定する必要はない
  - ◆ ②Type 2ドライバー / 32bitクライアント・インスタンス
    - クライアントインスタンスにて、DBをカタログする
  - ◆ ③Type 2ドライバー / 64bitクライアント・インスタンス
    - ネイティブ・ドライバー・パスと環境変数LIBPATHをlib32に設定する

ネイティブドライバーパス  
DB2UNIVERSAL\_JDBC\_DRIVER\_NATIVEPATH  
DB2\_JCC\_DRIVER\_NATIVEPATH

```
./home/db2inst/sqllib/db2profile
export LIBPATH=/home/db2inst/sqllib/lib32:$LIBPATH
```

環境変数 LIBPATH

1. setupCmdLine.sh(bat)に記述
2. WAS起動ユーザの.profileに記述
3. WASアプリケーション・サーバーのプロセス定義に設定



© 2012 IBM Corporation
152

DB2V9以降は、WindowsとLinuxを除き、DB2クライアントを含め64bit版のみの提供となりました。32bitのライブラリーもありますが、これは旧アプリを稼働させるために実行に最低限必要なものだけが入っているライブラリーになります。

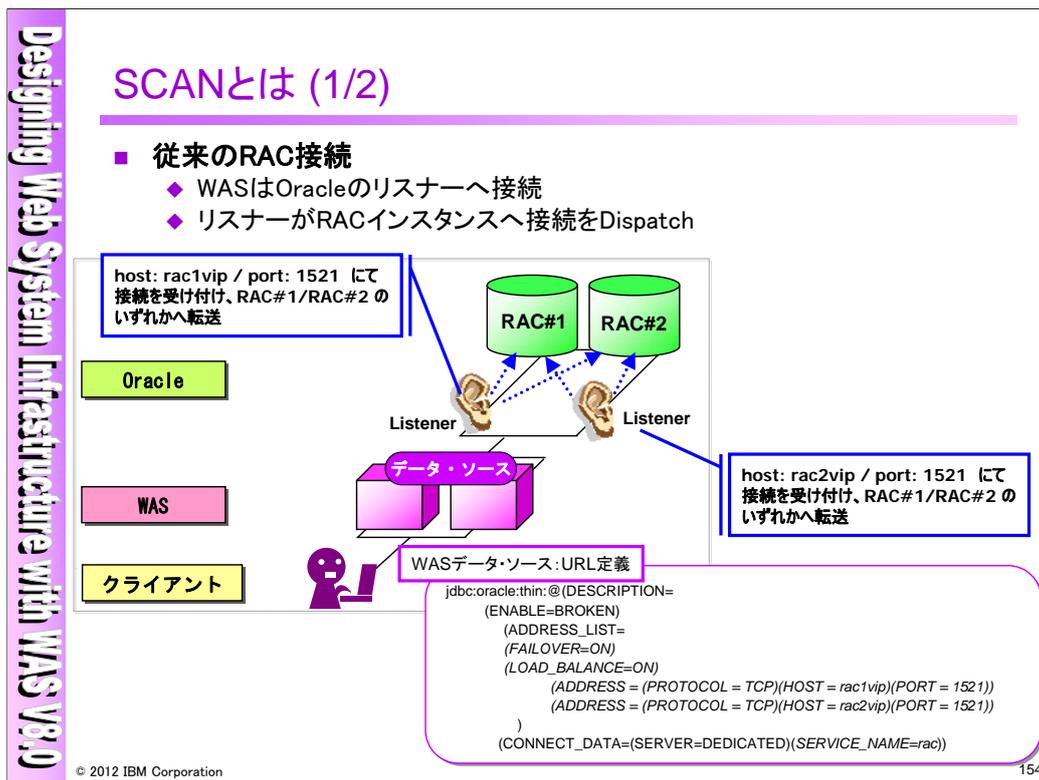
WASは32bit/64bitを選択でき、32bitWASから64bitDB2に接続するためには、環境により特別な設定が必要です。

- SCAN リスナーとは
- Oracle を WASセッションDBとして  
利用する場合のTips
- WAS-Oracle OS認証
- WAS-Oracle接続のタイムアウト指定

### 3. WAS-Oracle接続設計

- ・WAS-DB2接続設計
  - JDBCドライバー設計
  - データ・ソース設計
  - アプリケーション設計
  - パフォーマンス / 問題判別
  - Hints & Tips

[・WAS-Oracle接続設計](#)

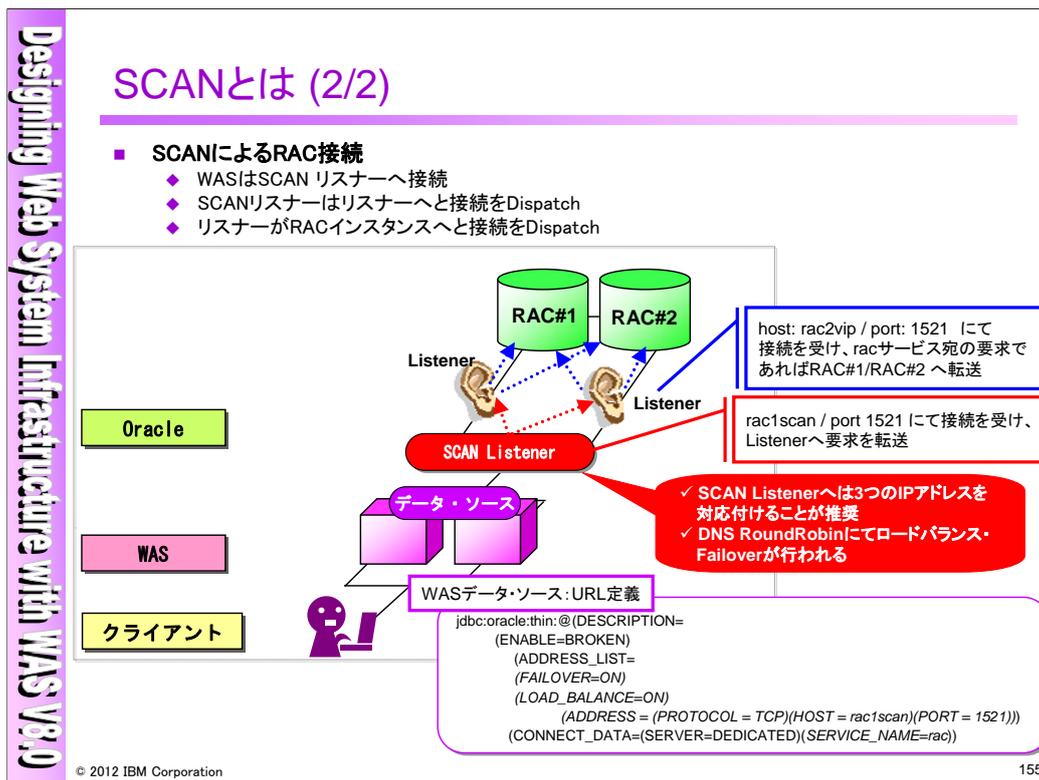


従来のWAS-OracleRAC接続においては、WASからはOracleリスナーに接続します。Oracleリスナーは通常、物理ノードにつき1プロセス起動してクライアントからの接続を待ち受けています。

OracleリスナーはOracleインスタンス宛のリクエストを受け取り、リクエストにて指定されたサービスを提供しているOracleのインスタンスにその接続を転送します。

この図では、耳の絵がリスナーをあらわし、リスナーからはRAC #1、RAC#2 のインスタンスに対して接続をディスパッチしています。

このページで接続している接続の流れ(クライアント→Oracleリスナー→Oracleサーバー)は、RACの場合も、non-RAC(シングル・インスタンス)の環境でも同じです。



SCANが利用される想定Oracle RAC サーバー群への接続は、こちらのページの図のようになります。

WASからはSCAN Listenerに接続します。

WASからのリクエストを受け取ったSCANリスナーは、各物理ノード上に存在するOracleリスナーに要求を転送し、

Oracleリスナーがさらにその要求を、宛先のサービスが稼働するOracle RACインスタンスへと転送します。

SCAN Listenerに紐付けるホスト名にはDNSラウンドロビンを利用して複数のIPアドレスにロード・バランスが行われるようになっています。

SCANのホスト名に対応づけるIPアドレスを、RACを構成する複数の物理ノードに跨る複数NICに割り当てられるように構成することで

SCANリスナーがSPOFとならない仕組みを実現しています。なお推奨として、SCANアドレスには3つのIPアドレスを紐付けることとなっています。

このようなSCANを利用する構成では、上記のような仕組みとなっていることから、WASからの接続先データソース定義としては1つのホスト名・ポート番号の組だけを指定すればOKです。

## OracleをWASセッションDBとして利用する場合のTips

- WAS セッション管理について
  - ◆ WASでは、セッションの永続化を選択する場合、セッション・オブジェクトをデータベースもしくはメモリーに保持することができる
  - ◆ WAS セッション・オブジェクトの保管先をOracle Databaseとした場合、セッション・テーブルには、格納するセッション・オブジェクトのサイズに応じて Small/Medium/Largeの3種類の列が定義される
  - ◆ デフォルト設定では Mediumサイズのセッション用に LONG RAW型の列が定義されるが、LONG RAW よりも BLOB型で利用するほうが性能上有利
- 【推奨】カスタム・プロパティ useOracleBLOB を作成し、値をtrue とする
  - ◆ WAS セッション管理のカスタム・プロパティとして設定
  - ◆ なぜ、デフォルトで有効になっていないのか？はNoteをご参照ください。

### 【背景】なぜデフォルトで useOracleBLOB が trueになっていないか？

Oracle 9i以前のバージョンのOracle JDBC THIN(Type4) Driverでは、BLOB型で4KB以上のデータを扱うことがサポートされていなかったため、デフォルトではLONG RAW 型列としてセッションテーブルが作成されるようになっています。

Oracle 10g以降のOracle JDBC Driverを使う場合では、THIN DriverにてBLOB型でデータを扱う上でのサイズ制限は撤廃されているため、BLOB型を利用する際のデメリットはありません。

Designing Web System Infrastructure with WAS V8.0

## WAS - Oracle OS認証

- Oracle OS認証
  - ◆ Oracleではユーザーのパスワードを持たず、OSにユーザー認証の制御を任せる機能
  - ◆ WASからの接続時には、getConnection引数として空文字列を渡す
    - ▶ Oracle接続ユーザはJVM起動引数として定義
  - ◆ DB接続ユーザーのパスワードをWAS側の構成ファイルやアプリケーション関連ファイルに書かなくてよいため、DB接続ユーザー・パスワードの変更が頻繁に行われるシステムにおいて運用上の利便性向上を目的として利用されることがある
  - ◆ 但し、TCP接続経由のリモートOS認証はセキュリティー上の理由から推奨されない
    - ▶ 詳細はOracleマニュアルを参照

**WAS**

JDBCアプリケーション

```
String userID="";
String pwd="";
conn=ds.getConnection(userID, pwd);
... (以下略)
```

userID, pwd  
ともに空文字列

**Oracle JDBC Driver**

**JavaVM**

Java起動引数

-Duser.name=**wasora**

Oracle接続ユーザ=**wasora**

**Oracle**

【事前準備】  
Oracle初期化パラメーターの変更

【事前準備】  
ops\$wasoraユーザー作成済み

wasoraユーザーの認証(OS認証)⇒OK

【事前準備】  
wasoraユーザー作成、パスワード定義済み

© 2012 IBM Corporation 157

Designing Web System Infrastructure with WAS V8.0

## WAS-Oracle接続タイムアウトに関わる設定

- 「タイムアウト」= 処理時間に一定の上限を設けておくこと
  - ◆ ある処理に時間がかり過ぎた場合に、処理を中断しエラーを返すことで、ユーザーに処理の失敗を通知する
  - ◆ タイムアウトがあることで、クライアントがサーバーからの応答を無限に待たずに済む  
⇒ 状況がわかる・時間がたったらリトライする等のアクションが取れる
- お客様要件にあわせて適切な設定値をご確認ください。
  - ◆ タイムアウトが長すぎると、クライアント(ブラウザ)への応答が長時間戻らない
  - ◆ タイムアウトが短すぎると、必要な処理が常に中断されエラーとなる

© 2012 IBM Corporation 158

タイムアウトは、サーバーの高負荷状態やネットワーク障害時などに、クライアントが無限に待つ事態を発生させないために必要な考え方です。

WASからDBへの接続中におけるタイムアウト時間を決めるための設定項目として、当ページの図に記したようなパラメーターがあります。

※Oracle部分については、弊社で調査した内容です。詳しくはOracle社へご確認ください。

Designing Web System Infrastructure with WAS V8.0

■□□ SQLの処理時間が想定範囲を超過している状況への対応 □□■

### Oracle: select for update 文のタイムアウト指定

- Oracleでは、select for update 文の実行時間に上限を設定することができる
  - ◆ 想定した以上長い時間、DBサーバーからの応答がない場合には処理を中断することで、クライアントを待たせすぎない制御ができる

例)

```
SELECT col1 FROM tab1 WHERE col1 = 1 FOR UPDATE OF col1 WAIT 10
```

select for update 文のタイムアウト指定

Webアプリケーション・サーバー

Application

Managed Connection

DataSource

Connection Pool

JDBC Driver

DataStoreHelper

DBサーバー

Data

Physical Connection

© 2012 IBM Corporation

159

Designing Web System Infrastructure with WAS V8.0

■□□ データベース・サーバー、ネットワーク障害などに起因する接続断への対応 □□■

### Oracle JDBCドライバー固有のパラメーター

- **oracle.net.CONNECT\_TIMEOUT (Type4のみ)**
  - ◆ データベースへの接続が確立できるまでに待つ秒単位の時間
  - ◆ 設定時間が経過すると、JDBCドライバーが接続をクローズ
  - ◆ デフォルトは0秒で、タイムアウト値はデフォルトのシステム・タイムアウト値となる
- **oracle.jdbc.ReadTimeout (Type4のみ)**
  - ◆ 接続ソケットの読み取りがタイムアウトになるまでのミリ秒単位の時間
  - ◆ アプリケーションで最長の照会を実行するのに必要な時間より数秒大きい値とする  
⇒ DB接続用ソケットにおける読み込み処理が開始されるまでに待つ時間とする
  - ◆ 注意点:長時間かかっているSQL処理(接続障害ではないもの)もタイムアウトとなる

© 2012 IBM Corporation 160

Designing Web System Infrastructure with WAS V8.0

■ □ □ データベース・サーバー、ネットワーク障害などに起因する接続断への対応 □ □ ■  
**OS KeepAlive関連パラメーターのチューニングについて**

- WAS-DB2接続同様に、OS KeepAlive設定を適切にチューニングする必要がある (⇒P147-148参照)
- WAS側で無効となったOracle接続を検出するための設定
  - ▶ デフォルトの設定では、プロセスダウン、ネットワーク障害時などWAS-Oracleサーバー間の通信が途切れた場合、WASがOracleサーバーからの応答を待ち続けてしまう
- ◆ URL定義の文字列に(ENABLE=BROKEN)を記述する
  - ▶ ENABLE=BROKENを記述することで、OSで設定したTCP KEEPALIVEの時間間隔でWASからOracleサーバーに対してprobeパケットが送信されるようになる
  - ▶ 接続障害が発生した時はORA-03113エラーを検出(⇒StaleConnectionException)

WASデータソース: URL定義

```
jdbc:oracle:thin:@(DESCRIPTION=
(ENABLE=BROKEN)
(ADDRESS_LIST=
(ADDRESS=(PROTOCOL=TCP)(HOST=rac1vip)(PORT=1521)))
(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=rac))
```

161

Oracleに接続する際も、WAS-DB2接続時と同様に、OS KeepAlive設定を適切にTuningする必要があります。

Oracleを使う場合のTipsとしては、クライアント(=WAS)側で無効となった接続を検出できるようにWAS側からprobeパケットを送信するための設定として、URL定義の文字列に(ENABLE\_BROKEN)を記述しておいてください。

Designing Web System Infrastructure with WAS V8.0

## DTPサービスの構成

- Oracle 10gでの2フェーズ・コミット処理への対応(11gでは不要)
  - ◆ DTP (Distributed Transaction Processing)サービスの構成
    - 分散トランザクション環境において、全てのブランチを特定のインスタンスで実行することを可能にするサービスである
    - 分散トランザクション間でのトランザクション・フェール・オーバーを可能にする
  - ◆ 考慮点
    - 全てのブランチを特定インスタンスで実行するため、片方のRACインスタンスに負荷が集中する可能性がある
  - ◆ 対応策
    - DTPサービスを複数構成し、処理を分散させる

2フェーズ・コミット処理は、Oracle RAC#1で優先して実行される

2フェーズ・コミット処理は、WAS1はOracle RAC#1を優先して実行する、WAS2はOracle RAC#2を優先して実行する

© 2012 IBM Corporation 162

DTPサービスを構成することで、分散トランザクション環境における全てのブランチを特定のインスタンスで実行することができます。また、トランザクション処理のフェール・オーバーも可能となります。ただし、片方のRACインスタンスに負荷が集中する可能性がありますので、ご注意ください。詳細な手順は、Oracle社のマニュアルをご確認下さい。

・Oracle RAC構成における2フェーズ・コミット時の考慮点 (WAS-08-013)

<http://www.ibm.com/jp/domino01/mkt/cnpages1.nsf/page/default-0002BA4D>

## Oracle 11gR2における32bit Client

- Oracle11gR2 には、\$ORACLE\_HOME/lib32が存在しない
  - ◆ 32bitのライブラリーはOracleデータベース・サーバー64bitとOracleデータベース・クライアントの64bit版のメディアには含まれない
- 32bitのクライアント(別メディア)を新しいORACLE\_HOMEにインストールして入手する

ワークショップ、セッション、および資料は、IBMまたはセッション発表者によって準備され、それぞれ独自の見解を反映したものです。それらは情報提供の目的のみで提供されており、いかなる参加者に対しても法的またはその他の指導や助言を意図したのではなく、またそのような結果を生むものでもありません。本講演資料に含まれている情報については、完全性と正確性を期するよう努力しましたが、「現状のまま」提供され、明示または暗示にかかわらずいかなる保証も伴わないものとします。本講演資料またはその他の資料の使用によって、あるいはその他の関連によって、いかなる損害が生じた場合も、IBMは責任を負わないものとします。本講演資料に含まれている内容は、IBMまたはそのサプライヤーやライセンス交付者からいかなる保証または表明を引き出すことを意図したもので、IBMソフトウェアの使用を規定する適用ライセンス契約の条項を変更することを意図したものでなく、またそのような結果を生むものでもありません。

本講演資料でIBM製品、プログラム、またはサービスに言及していても、IBMが営業活動を行っているすべての国でそれらが使用可能であることを暗示するものではありません。本講演資料で言及している製品リリース日付や製品機能は、市場機会またはその他の要因に基づいてIBM独自の決定権をもっていつでも変更できるものとし、いかなる方法においても将来の製品または機能が使用可能になると確約することを意図したものではありません。本講演資料に含まれている内容は、参加者が開始する活動によって特定の販売、売上高の向上、またはその他の結果が生じると述べる、または暗示することを意図したもので、またそのような結果を生むものでもありません。パフォーマンスは、管理された環境において標準的なIBMベンチマークを使用した測定と予測に基づいています。ユーザーが経験する実際のスループットやパフォーマンスは、ユーザーのジョブ・ストリームにおけるマルチプログラミングの量、入出力構成、ストレージ構成、および処理されるワークロードなどの考慮事項を含む、数多くの要因に応じて変化します。したがって、個々のユーザーがここで述べられているものと同様の結果を得られると確約するものではありません。

記述されているすべてのお客様事例は、それらのお客様がどのようにIBM製品を使用したか、またそれらのお客様が達成した結果の実例として示されたものです。実際の環境コストおよびパフォーマンス特性は、お客様ごとに異なる場合があります。

IBM、IBM ロゴ、ibm.com、DB2、Informix、Power、PuleScale、Tivoli、WebSphere、z/OSは、世界の多くの国で登録されたInternational Business Machines Corporationの商標です。

他の製品名およびサービス名等は、それぞれIBMまたは各社の商標である場合があります。現時点でのIBMの商標リストについては、[www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml)をご覧ください。

Linuxは、Linus Torvaldsの米国およびその他の国における登録商標です。

WindowsはMicrosoft Corporationの米国およびその他の国における商標です。

UNIXはThe Open Groupの米国およびその他の国における登録商標です。

JavaおよびすべてのJava関連の商標およびロゴはOracleやその関連会社の米国およびその他の国における商標または登録商標です。