# DB2 24時間365日ノンストップサービスの実現手法の比較評価

高谷 尚子　東 忠克　白井 徹哉

# Implementation of 24 Hours, Every Day Non-stop Service with DB2

Naoko Takaya　Tadakatsu Azuma　Tetsuya Shirai

企業の基幹業務を支え, 24時間365日稼働し続けるシステムでは, 災害への対策も必要不可欠である. 本論文では, 災害対策としてDB2®から提供される最新技術であるHADR（High Availability Disaster Recovery）とQ-Replicationに着目し, それぞれの機能や特徴を比較検討するとともに, 災害時における引き継ぎ／回復処理の考慮点を述べ, 両ソリューションのパフォーマンスやデータの保全性について比較評価する. また最後に, 両ソリューションを選択する上で指針となるQAフローを示す.

Disaster counter measure is essential for systems running 24 hours every day, supporting an enterprise's core business processes This paper focuses on two latest technologies, High Availability Disaster Recovery (HADR) and Q-Replication that are provided as DB2®'s disaster countermeasures, compares the features and functions of both solutions as well as switchover/recovery considerations at the occurrence of a disaster. Also, it describes the comparative evaluation based on the performance and data integrity. The QA-flow shown at the end should serve as guide for selection between these two solutions.

Key Words & Phrases：高可用性, 災害対策, HADR, Q-Replication, クライアント・リルート, データ保全性
high availability, disaster recovery, HADR, Q-Replication, client-reroute, data integrity

## 1. はじめに

2001年9月11日に発生した同時多発テロや日本における地震などの災害は, 情報システムを破壊し、企業活動に深刻な被害をもたらす可能性がある. 企業における情報システムの重要性が増すにつれ, このような脅威への対策が求められており, 実際にユーザーからの技術相談も増えている.

現在, データベースの高可用性構成としてIBMのHigh Availability Cluster Multi Processing（以降HACMP™と記す）のようなクラスター・マネージャーとRDBMSを組み合わせたものが広く採用されている. Oracle社のReal Application Cluster（RAC）も同様のソリューションに位置づけられる. しかし, このようなクラスター構成のDBサーバーは, サーバー間でディスクが共有されており, そのディスクが破壊されてしまった場合にはすべてのデータが失われてしまうため, 災害対策のソリューションとはなりえない.

提出日：2004年08月31日　再提出日：2005年1月18日

このような中, 複数サイトにサーバーとディスクを配置し, 災害に対応するための機能として, DB2 UDB™の最新のリリースであるV8.2でHigh Availability Disaster Recovery（以降HADRと記す）が登場した[1]. またWebSphere®MQのキューを使用してデータの高速なレプリケーションを行うQ-Replicationも登場した[2].

HADRとQ-Replicationの単純比較表は製品マニュアル[4, 5]などで見受けられるが, ニーズに合わせ, 二つの製品をどのように選択すべきかを判断するためのガイドラインとしては不十分である. そこで本論文では, 災害対策のニーズに対応するためにパフォーマンスやデータの保全性などの観点からHADRとQ-Replication の構成について比較評価した.

なお, アプリケーションによって更新されたデータを別サイトへ転送, 同期の取れたデータベースを保持するため, IBM Enterprise Storage Server®などのディスク装置が提供するリモート・コピーの仕組みを利用するソリューション[3]もあるが, 本論文ではソフトウェアの機能にて実現可能なソリューションに焦点を絞っ

て論ずることとする.

以下, 本論文では2章でQ-ReplicationとHADRの機能概要を述べ, 3章で両ソリューションの特長, 4章で障害発生時の運用について, 5章でQ-ReplicationとHADRのシステム性能評価の結果を示し, 最後, 6章で両ソリューションの選定基準を筆者らが作成したQAフローを元に述べる.

## 2. 機能概要

### 2.1 Q-Replicationの機能概要

Q-Replicationは, 従来のレプリケーション機能であるSQL-Replicationがレプリケーション対象表の変更履歴を収集するために中間表として用いていたChange Data表(以降CD表と記す)の代わりにキューを使用して, 表単位のレプリケーションを行う製品である[4]. Q-Replicationでは, Q-Captureと呼ばれるプログラムがDB2のログから変更データを収集し, キューにデータをメッセージとしてPUT(入力)する. ターゲット側ではQ-Applyと呼ばれるプログラムがキューからメッセージをGET(出力)し, ターゲット表へ変更を適用する(**図1**). また, リモートに対してレプリケーションを行う場合, MQのCHANNELおよびLISTNERを使用して, TCP/IP通信でキュー間のデータ受け渡しを行う.

Q-Replicationは, SQL-Replicationに比べ, スループットの向上やデータ鮮度の向上などを目的として開発された製品である. さらに, 複数サーバー間でのレプリケーションを行う方法が2種類登場した. 一つは2台のサーバー間で双方向レプリケーションを行うBidirectionalレプリケーション(以降Bidirと記す). もう一つは, 2台以上のサーバー間で互いにレプリケーションを行うPeer-to-Peerレプリケーション(以降P2Pと記す)である(**図2**).

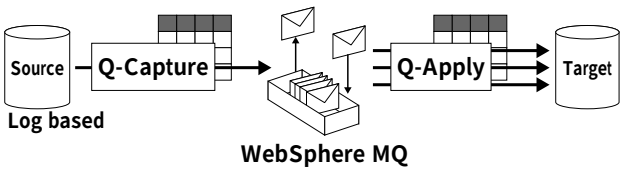


**図1. Q-Replicationのアーキテクチャー**(出典:[6]から抜粋後修正)

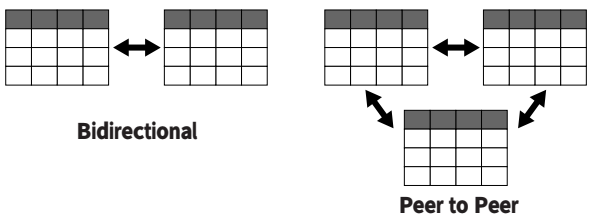


**図2. 複数サーバー間でのQ-Replication**(出典:[4]から抜粋後修正)

このようなレプリケーションの場合, データの二重更新を防ぐため, 適切な衝突検知方法も提供されており, どのデータをどこに反映させるかを時刻または, 設定に基づいて制御することができる. 詳しくは4.2.1項にて述べる.

### 2.2 HADRの機能概要

HADRは, あるデータベース(プライマリー・データベース)で発生した更新処理のログを, 別のデータベース(スタンバイ・データベース)にネットワークを経由して転送, 適用することで, 地理的に離れたシステム・センターに全く同一の内容を持ったデータベースを維持することのできる機能である. HADRはデータベース単位で設定され, 災害の発生などでシステム・センター全体が破壊されても, 業務サービスの継続提供が可能となる(**図3**).

HADRでは, プライマリーとスタンバイとを切り替えるテイクオーバー機能も提供している. TAKEOVERコマンドをスタンバイ側で実行すると, これまでのスタンバイ・データベースがプライマリー・データベースに切り替わり, プライマリー・データベースがスタンバイ・データベースに切り替わる.

### 2.3 クライアント・リルート機能

Q-Replication, およびHADRと組み合わせて使用できるDB2 UDB V8.2の新しい機能として, クライアント・リルート機能も提供される[1]. この機能では, Q-Replicationのターゲット・データベース, HADRのスタンバイ・データベースを, それぞれ障害発生時の代替データベースとして登録しておくことができる. これにより, Q-Replicationのソース・データベース, またはHADRのプライマリー・データベースで障害が発生した場合, それらに接続されたアプリケーションは, 代替データベースに自動的に再接続される. 代替データベースに接続先を切り替えられたアプリケーション

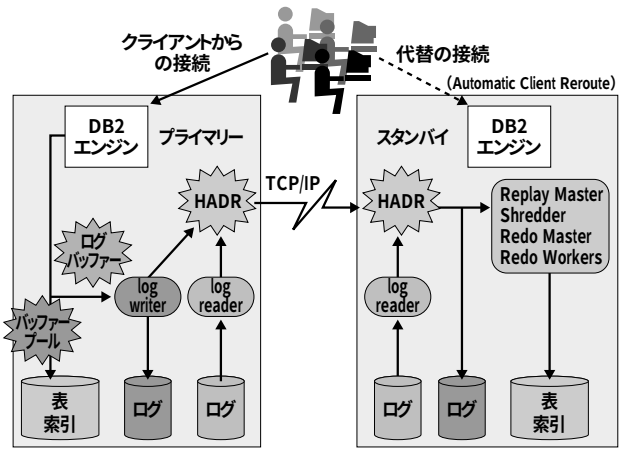


**図3. HADRのアーキテクチャー**(出典:[5]から抜粋後修正)

にはいったんエラーが返されるが, データベースへの再接続処理は不要であり, そのままSQLを継続実行できる.

## 3. Q-ReplicationおよびHADRを用いたソリューションの特長と制約

Q-ReplicationとHADRを比較すると, 実現可能な要件, 制約事項とも異なる[4, 5]. ここでは特に高可用性ソリューション, 災害対策ソリューションの実現という観点から双方を比較する.

### 3.1 システムの可用性

今日広く用いられているHACMPなどのクラスター・マネージャーを使った高可用性ソリューションでは, ディスクの引き継ぎやデータベースの整合性回復といった処理が必要となる[7]ため, 短時間での業務回復を実現させるための様々な考慮点があった[8]. Q-ReplicationやHADRのソリューションでは, 資源の引き継ぎやDB2の起動, 整合性回復処理を必要としないため, クライアントからのデータベース接続回復までの所要時間の大幅短縮が可能になる.

2.3節で示したように, Q-Replication, HADRのいずれにおいても, データベース・サーバーの障害発生時には, アプリケーションは, クライアント・リルートの機能を用いて代替サーバーのデータベースに接続, 業務を継続することが可能である.

Q-Replicationを用いたソリューションの場合, 代替サーバーとして登録するレプリケーション先のDB2は常に活動状態で, かつアプリケーションからも接続可能な状態である. 従って, クライアント・リルートの機能でサーバー障害を検知し, 代替サーバーへ自動再接続がなされるまでの時間が, データベースへの接続回復に必要な時間である. クライアントの側では, 応答待機時間を指定する変数の使用がV8.2より可能になった. この変数に設定された時間が経過してもサーバーからの応答がなければ, クライアント・リルートが起動される. サーバー障害を誤検知することを防ぐため, このレジストリー変数の設定値は, 通常のSQL応答時間よりも大きくする必要があるが, 例えば10秒と設定したならば, 約10秒程度での接続回復が可能である.

HADRを用いたソリューションの場合, HADRのスタンバイ・データベースは, 常に活動状態となってはいるが, アプリケーションからの接続はできない. TAKEOVERコマンドの実行により, スタンバイ・データベースをプライマリー・データベースに変換する必要がある. 従って, 業務再開までの所要時間は,「① プライマリー・データベースにおける障害検知の時間」+「② TAKEOVERコマンドを実行し, 完了するまでの時間」+「③ クライアントからの接続が代替サーバーへ確立される時間」となる. 上記①, ②の処理が継続している間に, クライアント側での応答待機時間の設定に基づいてサーバー障害が検知され, 代替サーバーへの再接続が繰り返し試行されることになる.

なお, HADRは, プライマリー・データベースでの障害を検知し, TAKEOVERコマンドを自動実行する機能を提供していない. 従って, この二つの処理をHACMPのようなクラスター・マネージャー製品を使用して実現するか, あるいは障害通知を受けた管理者が手動でTAKEOVERコマンドを実行するような運用を行うことになる.

業務再開までの所要時間という観点でQ-Replication, およびHADRのソリューションを比較してみると, TAKEOVERコマンドによる切り替えが不要なQ-Replicationの方が短時間での業務再開が可能であるといえる.

### 3.2 データの保全性

次に, データの保全性について比較してみることとする. DBサーバーに障害が発生, アプリケーションが代替サーバーへ自動再接続されたというケースを想定し, アプリケーションがそれまでに完了したトランザクションによる更新情報を正しく参照できるのか, という観点からQ-ReplicationとHADRによるソリューションを比較する.

#### 3.2.1 Q-Replicationにおけるデータの保全性

Q-Replicationは非同期でソース表からターゲット表, またはその逆方向へ変更を適用するため, 災害時にソース表とターゲット表でデータの整合性が合わない状況が発生することは避けられない. 例えば, ソース側が障害で停止した場合, DB2のログには書き出されているがまだQ-Captureが収集していないデータ, および, ソース側のキューにはPUTされているがまだターゲット側へ転送していないデータはターゲット側で適用することができないため, ソース表とターゲット表とでデータの不整合が生じる.

#### 3.2.2 HADRにおけるデータの保全性

HADRは, トランザクションが完了する度にログをプライマリーからスタンバイに転送するため, データ保全性に優れている. HADRでは, 同期(SYNC), 近同期(NEARSYNC), 非同期(ASYNC)の3種類のログ転送モードを提供している[5]. それぞれログの転送が完了したと認識するタイミングが異なるため, プライマリー・データベースの障害発生時にデータが失われてしまうリスクもそれぞれ異なる.

同期モードでは, 完了したトランザクションのログがプライマリーとスタンバイの双方のログファイルに書き込まれたことが確認できた時点で, アプリケーションにトランザクション完了の通知を返す. 従って, スタンバイ側のログファイルにまで確定済みの更新ログが書き込まれていることが保障される. プライマリー, スタンバイの双方のDB2が同時にダウンし, かつログファイルも破壊される, という状況が重ならない限りはデータを失うことはない.

ただし, 同期モードのHADRを使用した場合でも, スタンバイ側からログ書き込み完了の応答を受ける前にプライマリー・データベースがダウンしてしまうような状況は発生し得る. このとき, アプリケーションは, エラーを受け取り, トランザクションが正常完了しなかったと判断するが, スタンバイ側には更新内容が残されている. 同期モードでは, 確定されたトランザクションの更新がスタンバイ側に反映されることは保障できるが, サーバー障害のタイミングで実行, 失敗した更新処理の結果が残る可能性は否定できない点を注意したい.

近同期モードでは, スタンバイ側のDB2からログレコード受信の応答があった時点でアプリケーションにトランザクション完了の通知を返す. 近同期モードも高い保全性を提供し, プライマリー, スタンバイの双方のDB2が同時にダウンする, という二重障害が発生してしまうケースでなければデータは失われない.

非同期モードでは, ログ・データがTCP/IPスタックに渡され、ソケット送信呼び出しが正常に戻された時点でアプリケーションにトランザクション完了の通知を返す. そのため, 本当にスタンバイ側にログが無事到着したことは保障されず, プライマリーとセカンダリーの間のネットワーク障害が発生したケースであっても, 更新済みのデータがスタンバイに転送されていない状況が発生し得る.

このように, 採用する方式によって達成可能な保全性のレベルは異なる. 5章にて示すが, 性能面でのデメリットはあるものの, データの保全性が最重要な要件であるならば, HADRの同期モードを採用することになるだろう.

## 4. 障害発生時の運用

### 4.1 整合性確保

障害発生時に代替サーバーに切り替えたとしても, 必ずしも代替サーバー上のDBの内容が, 業務的に整合状態にあるわけではないという点は, Q-Replication, HADRの各モードのいずれを用いたとしても考慮すべき点である. つまり, 成功したトランザクションの更新が代替サーバーへは反映されていなかったり, 失敗したはずの更新が取り消されていなかったりする可能性がある.

このようにQ-ReplicationやHADRによる高可用性ソリューションを提供する場合, 障害発生後, アプリケーションへのデータベース接続は回復されたとしても, データベースが業務上の観点からは整合状態にはなっていない可能性がある. そのため, 別途アプリケーション側で整合性確保を行う等, そのような状況を許容するようなアプリケーション設計をする必要がある.

### 4.2 障害が発生したサーバーの回復

#### 4.2.1 Q-Replicationにおけるサーバーの回復

Q-Replicationでは, ソース側が停止した場合, 引き継ぎ作業としてソース表とターゲット表でデータの整合性を合わせるための運用がターゲット側で必要となる. その後ソース側が回復し, 業務が再開された場合, ソース側が停止していた間にターゲット側で行われた変更がソース表へ適用される. そのとき, 障害時にソース側で残っていたデータと, 回復後にターゲット側からソース表への変更データとの間で衝突が発生する. この衝突したデータをどう扱うかは, BidirかP2Pが使用されているかで異なる[4].

BidirではForceとIgnoreというモードがあり, Ignoreが設定されているサーバーが“勝者”となる. 例えば, 回復時にターゲット表からの変更をソース表へ適用したい場合, ソース側をForce, ターゲット側をIgnoreに設定する. この場合, ソース表からターゲット表への更新(ソース側で残されたデータ)と, ターゲット表からソース表への更新(その後の業務で変更されたデータ)で衝突が発生したときに, ターゲット表からソース表への更新がソース表でForce(強制)され, ターゲット側ではソース表からターゲット表への更新がIgnore(無視)される. 変更をどちらに反映したいかによりお互いの設定を適切に行う必要がある.

P2Pでは, タイムスタンプが使用される. 衝突したデータを適用するかしないかは, どのデータが最後に更新されたかを評価し, 常に最新のデータが反映される仕組みとなっている. しかし, そのデータがいつ変更されたかを管理するトリガーが行レベルに作成されるためBidirに比べてレプリケーション処理, 更新アプリケーションの双方のパフォーマンスに影響を及ぼす.

以上より, 複数サーバー間でレプリケーションを行う, または常に最新のデータが反映されるような業務においてはP2Pが向いている. 逆に, 衝突があまり発生せず, 2台のサーバー間での双方向レプリケーションを行い, パフォーマンスが重視される場合にはBidirが向いているといえる.

#### 4.2.2 HADRにおけるサーバーの回復

HADR構成におけるテイクオーバー発生後, 障害か発生した元のプライマリー・データベースが回復されると, 今度は現在のプライマリー・データベース(元のスタンバイ)からの更新ログをスタンバイ・データベースとして受け取ることになる. しかし, テイクオーバーを実行した時点で, プライマリー側に未転送のログが残ってしまっていたような場合, 新しいプライマリー・データベースではそのログが抜け落ちることになり, プライマリーとスタンバイとの間の不整合が発生する. この不整合状態のままでは, 新しいスタンバイはログを受け取ることができないため, 新しいプライマリー・データベースのフル・バックアップを新しくスタンバイとなるデータベースにリストアし, 両者の整合性を回復する必要がある.

## 5. Q-Replication, HADR使用時のシステム性能評価

### 5.1 更新処理の性能

Q-ReplicationやHADR構成を行った環境における, ソース・データベース側の更新処理の性能について, 実機にてデータの繰り返し挿入を行うテストを実行することで検証した. テスト内容としては, レコード長1016バイトの行を一行ずつコミットしながら, 10000行挿入, 完了までの時間を測定した. ケースとしては, HADRもQ-Replicationも行わない環境, Q-Replicationを行う環境, そして3種類のモードでのHADR環境, の5ケースを5回ずつ実行, その平均を取得した. 個々のケースにおけるテスト結果と相対比を表1に示す. DB2のパラメーターについては, バッファープールを1000にし, データベースのデータディスクとログディスクを別々のディスクに配置した. それ以外のDB2やWebSphereMQのパラメーターはすべてデフォルトである.

**表1**より, スタンバイ・サーバーと非同期に動作するHADRの非同期はHADRのない環境に比べ1%程度と更新トランザクションに与える影響が非常に小さいという結果が示された. またHADRの近同期はHADRのない環境に比べて約13%程度の処理時間への影響をもたらした. 13%程度であるならば, 今日使用可能な商用のサーバーで十分に補える程度のオーバーヘッドといえる. また, 同期モードでは, 約80%程度と非常に大きな影響を与える結果となったが, リモートサイトへのログの書き込みまで保障することを考えれば, やむを得ない結果といえる. 性能とデータの保全性とのどちらを重視するかによって選択は異なるであろう.

表1. 更新処理性能に与える影響

| | 処理時間(秒) | 相対比 |
|---|---|---|
| レプリケーションなし | 29.47 | 1.00 |
| Q-Replication | 29.67 | 1.01 |
| HADR-非同期 | 29.66 | 1.01 |
| HADR-近同期 | 33.43 | 1.13 |
| HADR-同期 | 52.77 | 1.79 |

一方, 更新トランザクションの処理とは独立してトランザクション・ログから更新情報を抜き出しているQ-Replicationでは, 1%程度と更新トランザクションに与える影響は非常に小さいという結果が示された.

なお, 今回のテストでは, ネットワークアダプターの受信プールサイズを調整した. 受信プールサイズが小さすぎるとパケットロスが発生し再送が行われる. これはHADRのパフォーマンスに大きな影響を与える. HADRではログをスタンバイ側へ転送する際にスタンバイ側を意識し実際にログが転送されたかどうかを確認する. もしスタンバイ側にログが転送されていなかった場合, スタンバイ側から再送の要求が送信されプライマリー側では要求のあったログを再度スタンバイ側へ送信する. そのため受信プールサイズが小さいと, 非同期モードにおいてもログの再送が発生しHADRのパフォーマンスに影響する.

一方, Q-Replicationにおいては受信バッファープールの影響は受けない. なぜならQ-Captureプログラムはログから収集したデータをMQのキューにPUTするのみで, その後はMQが責任を持ってターゲット側へ転送する. Q-Captureプログラムはターゲット側を意識せず, メッセージがターゲット側で受信されたかどうかは確認しない. しかし, ターゲット側ではソース側から送信されるメッセージを受信できる深さを持ったキューを用意する必要がある. 万が一, ターゲット側のキューが満杯になった場合, Q-Captureプログラムはそれ以上メッセージを送信できなくなるためエラーを出力し停止する.

### 5.2 Q-Replicationにおけるデータ鮮度

Q-Replicationでは, ソース側とターゲット側の処理が常に非同期でなされるため, ターゲット側のデータ更新に生ずる遅延を考慮する必要がある. Applyエージェントは, ソース側でキューにPUTされた変更データをターゲット側で即座にGETし, 複数のApplyエージェントで, 並列してターゲット表へ適用するため, 遅延時間が大幅に削減される. 実際にどの程度の遅延時間が発生するかを測定した結果を**図4**に示す. テストでは, 1秒間に約150件のデータが挿入される業務を想定し, データの平均遅延時間をApplyエージェントの数を変えながら計測した.

Applyエージェント数が1の場合, 一つのApplyエージェントでターゲット表への適用を行うため並列処理

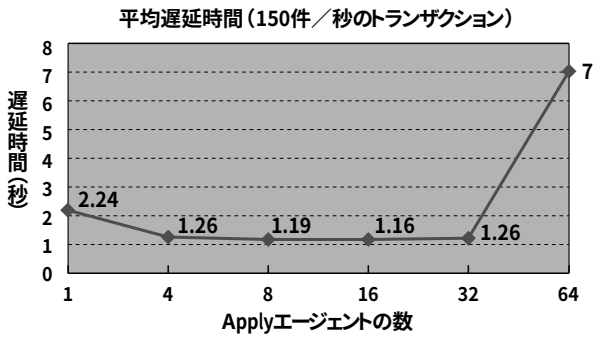


図4. Q-Replication使用時の遅延時間評価

が行われない. その結果, 他のApplyエージェント数の場合に比べデータの平均遅延時間が遅くなっていることが図4より確認できる. また, 逆にApplyエージェント数が64と多い場合, お互いのApplyエージェントがシステムのリソースを奪い合い, またターゲット表へのロックなどが発生しロック待ちが多発するためデータの平均遅延時間が遅くなる. 図4からもそのような結果が確認できる.

以上より, Applyエージェント数は必ずしも多ければ多いほどデータの遅延時間が短縮されるわけではないことが分かる. しかし, Applyエージェント数を調整することで, 遅延時間を平均約1秒に抑えることも可能である.

さらに, データの鮮度が高いということはそれだけ障害時に失われるデータ量が少ないといえる. 結果, Q-Replicationは災害対策の1ソリューションとして機能することも可能であることが確認できた.

なお, Applyエージェント数をチューニングする際には, 省略値の16から開始し, モニタリングにより数を調整することが製品マニュアルでも推奨されている[9].

今回の性能テストは, 以下のようなRS/6000® SP®の2ノード構成にて実施した.

モデル　：RS/6000 SP Silver Wide Node
CPU　：375MHz x 4
メモリ　：1GB
ディスク　：7133-T40（8GB x 2）
ネットワーク：SP-Switch（150MB/s）

## 6. Q-ReplicationおよびHADRの選定基準

今まで述べたとおり, HADRとQ-Replicationにはそれぞれ異なった特長と考慮点があり, どちらかが他方より優れているといったものではない. それぞれの特長を生かし, サービスの要件および環境に適したソリューションを提案することになる. そこで筆者はその際の指針となるQAフローを**図5**のように作成した.

QAフローに示したように, 確定されたトランザクションの更新データが確実にスタンバイ側にも記録されることを保障する必要があるのかどうか, または二つのサーバーを共にアクセス可能である必要があるのかどうか, といった要件が, 機能選定をする上で重要な項目となる.

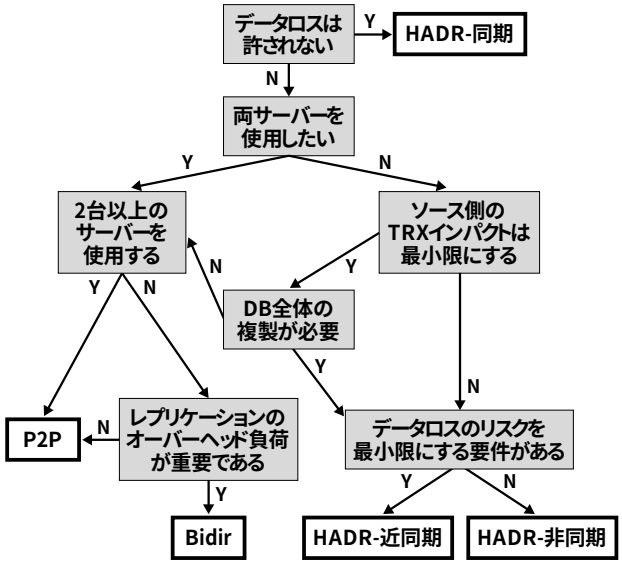


図5. QAフロー

また, 世の中の実際のアプリケーションを見てみると, 刻々と変化する情報を照会するアプリケーションなどで, 必ずしも常に最新のデータにアクセスできなくとも, 数秒前のデータであれば十分であるシステムは多い. そのようなシステムであれば, Q-Replicationにおけるターゲット側へのデータ反映の遅延について全く問題がないであろう.

なお, このQAフローはあくまでも一例であり, すべての状況に当てはまるわけではない. また, 今後Q-ReplicationもHADRも機能強化が予定されているため, それらの強化に伴ってこのQAフローも変化してゆくであろう.

## 7. まとめ

本論文では, 災害対策として, HADRとQ-Replicationを使用したソリューションを紹介した. それぞれ, 様々なサービス要件に応じた有効な機能や考慮点があり, 必ずしも一つのシステム構成だけですべてには対応できないことを述べた. そのために本論文では, 業務の用途に応じて適切な構成を柔軟に選択できるようQAフローを作成した.

災害時にスタンバイ機に自動再接続するDB2 UDB V8.2の新機能クライアント・リルートと合わせ, 今後HADRやQ-Replicationを用いた災害対策用データベースシステム構成を採用する企業が増えることが期待される. システムが導入される前に, 筆者らが本論

文で検証した運用における考慮点やQAフローが役に立てば幸いである.

謝辞

本論文を執筆するにあたり, ISE, データシステム部の皆様より, 多くの貴重なご助言をちょうだい致しました. この場を借りてお礼申し上げます.

参考文献

[1] Paul C Zikopoulos, That Bee in My Bonnet is IBM DB2 UDB Stinger, IBM developerWorks, 2004, http://www-106.ibm.com/developerworks/db2/library/techarticle/dm-0404zikopoulos/0404zikopoulos.pdf

[2] *IBM DB2 Information Integrator Introduction to Replication and Event Publishing,* GC18-7567-00, pp. 21-46, 2004

[3] Cathy Warwick et al., Implementing ESS Copy Services in Open Environments, IBM Redbooks, http://www.redbooks.ibm.com/redbooks/pdfs/sg245757.pdf, 2004

[4] *IBM DB2 Information Integrator Replication and Event Publishing Guide and Reference Version8.2,* SC18-7568-00, pp117-144, 2004

[5] *IBM DB2 Universal Database Data Recovery and High Availability Guide and Reference,* IBM, SC09-48313-01, pp.173-208, 2004

[6] Anthony J Ciccone, Beth Hamel, *The Next Generation: Q Replication, DB2 Magazine,* Quarter 3, Vol.9, Issue 3, pp22-24, 2004

[7] 室住正晴, CICS-UDB高性能基幹OLTPシステム構築事例, ProVISION Fall 2003 No.39, pp. 82-91

[8] IBM Software Group Toronto Laboratory, Delivering Ten Second Failover for High Volume Transactional Telco Applications with IBM DB2 Universal Database V8.1, 2003, ftp://ftp.software.ibm.com/software/data/pubs/papers/10sfailover.pdf

[9] *IBM Information Integrator Tuning for Replication and Event Publishing Performance,* SC18-9289-00, pp.19-20, 2004

**日本アイ・ビー・エム システムズ・エンジニアリング株式会社**
**インフォーメーション・マネージメント**

**髙谷 尚子** Naoko Takaya

**[プロフィール]**

**2003年日本アイ・ビー・エム システムズ・エンジニアリング株式会社入社. リレーショナルデータベース製品であるDB2 Universal Database for Linux, UNIX and Windowsの技術サポートを担当. 主にInformation Integratorやレプリケーション機能に関するお客様プロジェクトの技術支援や研修の実施, 解説資料の執筆などを実施している.**

**NTAKAYA@jp.ibm.com**

**日本アイ・ビー・エム システムズ・エンジニアリング株式会社**
**インフォーメーション・マネージメント**

**東 忠克** Tadakatsu Azuma

**[プロフィール]**

**日本IBM1986年入社. 92年に日本で初めてDB2-SQL/DSのDRDA構成のテストを実施後, 95年にISEへ出向. マルチプラットフォーム, 分散データベース上で稼動するDataPropagatorやInformation Integratorを中心にそれらの普及の為, 多くのお客様に技術支援を実施してきた.**
**これらの経験を生かし現在海外DB2 UDB開発部門へ次期バージョン等の開発支援も実施している.**

**OZUMA@jp.ibm.com**

**日本アイ・ビー・エム システムズ・エンジニアリング株式会社**
**インフォーメーション・マネージメント**

**白井 徹哉** Tetsuya Shirai

**[プロフィール]**

**1992年日本IBM入社. 96年のDB2/6000 Parallel Edition V1以降, UNIX、Windows環境でのDB2の普及を目指し, DB2製品の技術支援をIBM社内およびお客様に対して実施している.**
**著作に「DB2 UDB V7.1 Performance Tuning Guide(邦題:DB2 UDBパフォーマンス・チューニングガイド)」などあり.**

**tetsuya@jp.ibm.com**