WHITE PAPER

Mobile Linkを使用した統合データベースへのデータ移植および再移植

概要

本書では、ローがすでに同期済みの場合であっても、リモートのAdaptive Server Anywhereのすべてのローを統合データベースと同期させるためのテクニックについて説明します。これは、データを新しい統合データベースに入力する場合、またはリモート・データベースのデータを使用して既存のデータベースにデータをリカバリする場合に便利なテクニックで、リモート・データを同期していない変更としてマーク付けしてリモートのAdaptive Server Anywhereのアンロードと再ロードを実行し、その後で統合データベースへのリモート・データの更新または挿入を行う特別な同期スクリプトを使用しします。

e-BUSINESS ANYWHERE

目次

統合データベースへのデータの移植や再移植が必要な場合

リモートの Mobile Link クライアントのデータベース内のデータを使用して、Mobile Link の統合データベースへのデータの移植または再移植の必要性があるのは、以下のような場合です。

事例 1：統合データベースが消失してしまい、バックアップがない。

- これは、当然回避すべき状況です。分散環境 (Mobile Link または SQL Remote) では、統合データベースの障害に対する予防が必要です。ハードウェアの問題 (ドライブのクラッシュ)から、データベースの破壊、誤ったバックアップとリカバリの手順などが原因で、障害は発生する可能性があります。

事例 2：フランチャイズ店を 1 度に 1 店舗ずつ追加している。

- データベースを 1 週間に 1 つずつ Mobile Link 設定に追加すると仮定します。統合データベース側には必ずしもなくてもよいデータが店舗ごとに存在しています。
  - 全店で共通の製品リスト
  - その店舗だけで発生する注文
- この例では、レコードのプライマリ・キーが他の店舗のプライマリ・キーと競合しないものとします。

事例 3: リモート・データベースには固有のデータがあり、統合データベースは単にレポート・センタとして使用されている。このシナリオでは、

- 統合データベースはデータの収集とレポート目的で使用されていて、そのすべてのデータがリモート・データベースから収集されています。
- 統合データベースが消失した場合でも、データを管理せず単に集約しているだけであるため、特に問題にはならないため、このような事例では、統合データベースをいつでも再構築できます。

使用可能なソフトウェアのバージョン

ここでは、Mobile Link と Adaptive Server Anywhere リモート・データベースの双方のバージョンが 8 であることを想定しています。このテクニックでは、Ultra Light のリモート・データベースは使用できません。また、バージョン 7.0.3 以降でもこのテクニックを使用できますが、バージョン 7 を使用する場合は、テクニックは同じでも、Mobile Link の SQL 文は異なります。たとえば、バージョン 7 では CREATE SYNCHRONIZATION DEFINITION 、バージョン 8 では CREATE SYNCHRONIZATION SUBSCRIPTION/CREATE PUBLICATION を使用してください。

要件

- 通常 Mobile Link では、統合データベースを 1 つ設定します。統合データベースとしては、Adaptive Server Anywhere、Sybase Adaptive Server Enterprise、Microsoft SQL Server、Oracle、IBM DB2 のいずれかを使用することができます。

- リモートには Adaptive Server Anywhere を 1 つ以上使用できます。

- Mobile Link では、同期ロジック用にスクリプトのバージョンを少なくとも 1 つ定義し、このバージョンを一連の同期スクリプトを定義するのに使用します。Mobile Link サーバは、異なるバージョンのスクリプトを実行するリモート・データベースを同時に同期させることができ、またアプリケーション自体は、複数のバージョンのスクリプトを使用できるので、たとえば、スクリプトのバージョンとして、1.0.0、2.0.0、またはその中間バージョン 1.5.0 を使用することができます。また、Salesperson、Support、Manager というようなバージョンも使用できますが、この場合は、リモート・データベースで 2 つの異なるスクリプト・バージョンを使用します。

- 2 ページの事例 1 「統合データベースへのデータの移植や再移植が必要な場合」で説明したように、Mobile Link を使用する設定がすでにある場合でも、以下の手順を実行できます。

  ここで説明する手順は、一度だけ実行するためのもので、統合データベースを再構築した後は、通常の手順を使用してください (つまり、再構築前と同じ手順を使用します)。

手順例

基本的には、ロードがログ・ファイルに記録されるよう、リモート・データベースのアンロードと再ロードを実行します。これにより、すべてのデータがまだ同期されていない挿入としてマーク付けされます。この後に同期を行うと、新規のプライマリ・キーを持つローが挿入されます。重複するプライマリ・キーを持つローは、使用しているビジネス・ロジックに応じた方法で処理されます。

以下の項では、手順の詳細について説明します。これらの同期スクリプトを通常の同期スクリプトから区別するために、新しいスクリプトのバージョンを使用してください。

統合データベースでの手順

最初にしなければならないのは、統合データベースにローがすでに存在しているケースの処理が必要かどうかの判断です。全リモート・データベースにおけるローのプライマリ・キーがそれぞれ異なり、それらのプライマリ・キーを持つローが統合データベースには存在しないことが明かである場合を除き、上記のケースの処理が必要になります。さもなくば、リモート・データベースからのアップロードを行った場合に、統合データベース内の既存のローと一致するプライマリ・キーのついたローの挿入が試みられてしまいます。

注意 同期ロジックが通常の同期ロジックとは異なる場合、その同期ロジックには新しいスクリプト・バージョン名を使用してください。以下の項で説明している例では、REBUILD というバージョン名を使用しています。

- 他のリモート・データベースまたは統合データベース内のローと同じプライマリ・キーを持つローがリモート・データベースにないことがわかっている場合は、通常の同期スクリプトを使用できます。この場合は、8 ページの「リモート・データベースでの手順」に進んでください。

- プライマリ・キーが一致するローを処理する必要があり、統合データベースが Adaptive Server Anywhere 8.0.2 以降である場合は、通常の同期スクリプトを使用することができます。ただし、upload_insert スクリプトで ON EXISTING 句を指定して INSERT 文を使用し、既存のローを処理する必要があります。この場合は、4 ページの「INSERT 文での ON EXISTING 句の使用」に進んでください。

- それ以外の場合は、強制的な競合解決を使用して競合を効率的に処理する新しいスクリプトを使用します。この場合は、5 ページの「強制的な競合解決」に進みます。

INSERT 文での ON EXISTING 句の使用

統合データベースが Adaptive Server Anywhere バージョン 8.0.2 以降の場合は、upload_insert スクリプト内の INSERT 文で ON EXISTING 句を使用することができます。ON EXISTING 句を指定することで、テーブルにすでに存在するローに対して以下のいずれかのアクションを選択することができます。

- キーの値が重複する場合にエラーを生成：これは、ON EXISTING 句を指定しない場合のデフォルトの動作です。

- エラーを生成せずに、入力ローを無視：この動作を指定するには、ON EXISTING IGNORE を使用します。

- 入力ローの値で既存のローを更新：この動作を指定するには、ON EXISTING UPDATE を使用します。

これらのアクションのいずれも適さない場合は、ON EXISTING 句を使用せずに、強制的な競合解決を使用する必要があります。

強制的な競合解決については、5 ページの「強制的な競合解決」を参照してください。

ON EXISTING 句の例

次のテーブルはリモート・データベースから同期されていると仮定します。

```
CREATE TABLE Product (
    prod_id          INTEGER NOT NULL PRIMARY KEY,
    price            INTEGER,
    prod_name        VARCHAR(30),
    )
```

Product テーブルには、イベントが 1 つだけ定義されています。

upload insert (Script Version = REBUILD)

```
INSERT INTO Product( prod_id, price, prod_name )
ON EXISTING UPDATE VALUES (?, ?, ?)
```

上記の INSERT 文は、統合データベースに存在しない新しいローの場合はそのローを挿入し、統合データベースにすでに存在する同一のプライマリ・キーを持つローの場合はローを更新します。

強制的な競合解決

Mobile Link の競合解決機能により、リモート・データベースからアップロードされるデータを統合データベースに柔軟に適用することができます。強制的な競合解決の使用方法の説明に入る前に、Mobile Link の競合について理解しましょう。

- 競合について

  競合は、リモート・データベースを最後に同期した後に、統合データベースで変更があったレコードをリモート・データベースがアップロードすると発生します。デフォルトでは、Mobile Link は、アップロードされたレコードを受け入れることで競合を解決します。またデフォルトの競合解決処理を無効にして、リモート・データベースからどの値を使用するのかをプログラムで制御することもできます。競合解決は、UPDATE でアップロードされたローにだけ影響しますが、Mobile Link の競合解決処理では、挿入と削除は対象にはなりません。

- 競合の強制解決について

  統合データベースの再構築を行うには、競合の強制解決という Mobile Link の機能を使用します。競合の強制解決を使用して、リモート・データベースからアップロードされたすべてのローを Mobile Link で処理できます。通常の競合解決とは異なり、競合の強制解決は挿入、更新、削除に対しても使用できます。統合データベース側で記述する Mobile Link スクリプトにより、各テーブルで挿入または更新のどちらを実行するのかを指定することができます。さらに、ロー単位でデータの処理を指定できます。

  競合の強制解決を使用する場合は、リモート・データベースのローが統合データベースに挿入されます。リモート・データベースでの処理が更新または削除の場合でも、ローは挿入されます。

強制的な競合解決の使用

使用している統合データベースの種類に関係なく、Mobile Link の競合の強制解決モードを使用して、アップロードされたデータを統合データベースに適用し、統合データベースにすでに存在するローと一致するプライマリ・キーを持つローを処理できます。ここでは、この処理の概要について説明します。

新しいスクリプトのバージョン名を使用して、次に挙げる各テーブルのテーブル・イベント用にスクリプトを作成します (他のテーブル・イベントにはスクリプトを作成しないでください)。

- begin_upload (Script Version = REBUILD)　このスクリプトを使用して、アップロードされたデータ用に統合データベースにテンポラリ・テーブルを作成します。

- upload_new_row_insert (Script Version = REBUILD)　このスクリプトを使用して、ローをテンポラリ・テーブルに挿入します。すべてのローがテンポラリ・テーブルに挿入された後に、end_upload テーブル・イベントが呼び出されます。

- end_upload (Script Version = REBUILD)　このスクリプトを使用して、テンポラリ・テーブルのすべてのローを永久テーブルに移動します。永久テーブルに存在しないプライマリ・キーを持つローの場合は、そのローを挿入します。それ以外の場合は、永久テーブル内のローを更新します (または、指定された競合解決を実行します)。すべてのローが永久テーブルに移動した後は、テンポラリ・テーブルを削除します。

  resolve_conflict スクリプトの代わりに end_upload スクリプトを使用することで、テンポラリ・テーブルにアップロードされたすべてのローを処理できます。1 つのテーブルに対して 1000 個のローがアップロードされた場合は、end_upload イベントは 1 度だけ呼び出されますが、resolve_conflict イベントは 1000 回呼び出されます。

強制的な競合解決の使用例

次のテーブルはリモート・データベースから同期されていると仮定します。

```
CREATE TABLE Product (
    prod_id             INTEGER NOT NULL,
    price               INTEGER,
    prod_name           VARCHAR(30),
    PRIMARY KEY( prod_id )
)
```

統合データベース上の対応するテーブルには、多くの場合は別のカラムがあります (以下を参照)。

```
CREATE TABLE Product (
    prod_id INTEGER     NOT NULL,
    price               INTEGER,
    prod_name           VARCHAR(30),
    last_modified       DATETIME DEFAULT TIMESTAMP,
    PRIMARY KEY( prod_id )
)
```

last_modified カラムは、統合データベースでだけ使用されます。このカラムは、リモート・データベースが最後にローをダウンロードした以降にローが変更されたかどうかを判定するためだけに使用されます。これにより、リモート・データベースにすでにローがあり、そのローが変更されていない場合に、リモート・データベースにロー

が送信されることを禁止します。Product テーブルの download_cursor は、このカラムを参照する WHERE 句を使用します。

Product テーブルには、以下のイベントが定義されています。

- begin_upload (Script Version = REBUILD)
- upload_new_row_insert (Script Version = REBUILD)
- end_upload (Script Version = REBUILD)

begin upload (Script Version = REBUILD)

```
CREATE TABLE #Product (
    prod_id             INTEGER NOT NULL PRIMARY KEY,
    price               INTEGER,
    prod_name           VARCHAR(30),
    PRIMARY KEY( prod_id )
    )
```

この begin_upload スクリプトは、リモート・データベースからアップロードされたすべてのローを格納するためのテンポラリ・テーブルを作成します。

このテーブルには、last_modified カラムがありません。リモート・データベースにはこのカラムがないため、ローのアップロード先のテンポラリ・テーブルにはこのカラムは不要です。

上記の構文は、Microsoft SQL Server、Sybase Adaptive Server Enterprise、Oracle 8i、Adaptive Server Anywhere で使用できますが、このテクニックをこれらとは異なる RDBMS で使用する場合は、RDBMS のマニュアルでテンポラリ・テーブルの作成および参照方法を確認してください。

upload new row insert (Script Version = REBUILD)

```
INSERT INTO #Product( prod_id, price, prod_name )
VALUES (?, ?, ?)
```

この upload_new_row_insert スクリプトは Product テーブル用に作成されたものですが、実際には #Product テンポラリ・テーブルにデータを挿入します。これは、Mobile Link の機能の 1 つで、名前およびカラム名が統合データベースでのものと一致する必要はありません。このようになっているのは、読みやすさのためで、重要なのはカラムのアップロード順であって、カラム名ではありません。

end upload (Script Version = REBUILD)

```
CALL sp_Product_insert_update();
DROP TABLE #Product
```

この end_upload スクリプトは、sp_Product_insert_update ストアド・プロシージャを呼び出します。このプロシージャは、挿入と更新を実行します (以下を参照)。

次に、このスクリプトはテンポラリ・テーブルを削除します。この時点で、sp_Product_insert_update ストアド・プロシージャが統合データベース内の実際の Product テーブルを更新したため、テンポラリ・テーブルは不要になります。

Adaptive Server Anywhere 統合データベースを使用している場合は、#Product テーブルの代わりにグローバル・テンポラリ・テーブルを使用できます。つまり、begin_upload_event を DELETE 文に置換し、ストアド・プロシージャだけを呼び出すように end_upload イベントを変更します。

グローバル・テンポラリ・テーブルの使用については、『Adaptive Server Anywhere リファレンスマニュアル』の「CREATE TABLE statement」を参照してください。

各テーブルに必要なイベントは、これですべてです。

sp_Product_insert_update ストアド・プロシージャは、以下のようになっています。

```
CREATE PROCEDURE sp_Product_insert_update()
BEGIN
    -- First insert all records that are in the temporary table
    -- that currently do NOT exist in the permanent table
    INSERT INTO Product( prod_id, price, prod_name )
    SELECT prod_id, price, prod_name
    FROM #Product t_p
WHERE t_p.prod_id IS NOT IN (
    SELECT prod_id
    FROM Product p2
    WHERE p2.prod_id = t_p.prod_id
);
-- Second update all records that are in the Product table
-- that currently DO exist in the #Product table
-- with the data that is in the temporary table
UPDATE Product
SET price = t_p.price, prod_name = t_p.prod_name
FROM #Product t_p
WHERE Product.prod_id = t_p.prod_id;
END
```

ストアド・プロシージャの 2 番目の部分と、統合データベースに既存のローへのアップロードを処理するユーザ定義のビジネス・ロジックと置換したいこともあるかもしれません。上記の例では、アップロードされたローで既存のローが置換されます。

リモート・データベースでの手順

統合データベースを設定した後は、リモートの Adaptive Server Anywhere データベースを設定し、そのデータをアップロードできます。Adaptive Server Anywhere 用の Mobile Link クライアント (dbmlsync) は、データベースが最後に同期された後に変更されたローだけをアップロードします。これは、Adaptive Server Anywhere のパブリケーション (バージョン 7 では同期の定義) に関係するテーブルのトランザクション・ログをスキャンすることで実行されます。リモート・データをアップロードするには、リモート・データベースのトランザクション・ログにローが含まれていることを確認する必要があります。これにより、dbmlsync が次に同期を実行するときに、データが (挿入として) 統合データベースにアップロードされます。

これを実現するには、特別なスイッチを指定して dbunload ユーティリティを使用し、リモート・データベースを再構築するのが最も簡単です。

Mobile Link の同期では、リモートの Adaptive Server Anywhere で以下の項目が必要です。

1. **パブリケーション：**リモート・データベースと Mobile Link サーバとの間で同期させるテーブルを指定します。

2. **同期ユーザ：** Mobile Link サーバに対応するリモート・データベースを一意に特定します。任意の値を使用できますが、2 つのリモート・データベースに同一の同期ユーザ名を設定することはできません。

3. **1 つまたは複数の同期サブスクリプション：**同期ユーザが同期させるパブリケーションの指定および同期オプションの指定を行います。通信プロトコル、Mobile Link サーバのホスト・アドレス、使用する同期スクリプトのバージョン名などをオプションとして指定します。

Mobile Link の同期を開始して間もない場合は、リモート・データベースではまだこれらの項目が定義されていません。リモート・データベースですでに Mobile Link の同期を使用している場合は、これらの項目がすでに定義されていることになります。

リモート・データベースの再構築およびすべてのデータのアップロードの例

以下のパブリケーション、同期ユーザ、同期サブスクリプションがリモート・データベースで使用されているものとします。

```
CREATE PUBLICATION Product(
         TABLE Product
);

CREATE SYNCHRONIZATION USER store31;

CREATE SYNCHRONIZATION SUBSCRIPTION TO "Product"
    FOR store31
    TYPE 'tcpip'
ADDRESS 'host=localhost;port=2439'
     OPTION ScriptVersion ='v1.0';
```

以下の手順は、リモート・データベースを再構築し、同期を実行して、Product テーブル内のすべてのローを挿入としてアップロードします。

1. 既存の統合データベースのリカバリを行う場合は、ユーザを削除して再追加し、統合データベースでの同期の進行の値をリセットします。

   - リモート・データベースを同期させると、Mobile Link はリモートの Adaptive Server Anywhere データベースが最後に同期に成功したログ・オフセットを格納します。そしてAdaptive Server Anywhere データベースを再構築すると、このログ・オフセットは無効になります。

   - DBMLUSER ユーティリティを使用して、ユーザを削除し、(統合データベースの) Mobile Link のシステム・テーブルにユーザを再追加します。

   ```
   dbmluser -d -u store31 -c "dsn=cons"
   dbmluser -u store31 -c "dsn=cons"
   ```

2. リモート・データベースのスキーマをアンロードします。

   - データをリモート・データベースに再ロードする前にパブリケーションとサブスクリプションを設定する必要があるため、スキーマはデータとは別にアンロードする必要があります。これにより、DBMLSYNC は INSERT を統合データベースにアップロードします。

   - -n スイッチを指定して dbunload ユーティリティを使用し、スキーマだけをアンロードします。

   ```
   dbunload -y -c "dsn=sales" -n -r newsales.sql
   ```

3. リモート・データベースからデータをアンロードします。

   - -d と -ix のスイッチを指定して dbunload ユーティリティを使用し、データだけをアンロードして、再ロードを外部的に実行するように指定します。外部的に再ロードを実行すると、データの再ロードはログ・ファイルでは INSERT として記録されます。次に例を示します。

   ```
   dbunload -y -c "dsn=sales" -ix -d -r newsales_data.sql c:\unload
   ```

4. 新しいリモート・データベースを作成し、スキーマを再ロードします。

   ```
   dbinit newsales.db
   dbisql -c dsn=newsales read newsales.sql
   ```

   これは、スキーマをもう一度作成します。同時に、dbunload の実行時に存在していたパブリケーション、同期ユーザ、サブスクリプションも作成します。

5. 次に、同期を設定します。手順は、リモート・データベースのアンロード時にすでに同期設定があるかどうかによって異なります。

   同期設定がすでに定義されている場合は、同期を削除して再定義し、同期の記録情報をリセットするだけです。

この例のデータベースの場合は、newsales データベースにログインし、以下の SQL コマンドを実行します。CREATE SYNCHRONIZATION SUBSCRIPTION 文は、*newsales.sql* ファイルからコピーできます。

```
DROP SYNCHRONIZATION SUBSCRIPTION TO "Product"
            FOR store31;

CREATE SYNCHRONIZATION SUBSCRIPTION TO "Product"
    FOR store31
    TYPE 'tcpip'
ADDRESS 'host=localhost;port=2439'
    OPTION ScriptVersion ='v1.0';
```

同期設定が定義されていない場合は、同期するテーブル用に通常のパブリケーション、同期ユーザ、同期サブスクリプションを定義する必要があります。以前に存在した統合データベースにデータを再移植する場合は、パブリケーションがすでに定義されていることがあります。

```
CREATE PUBLICATION Product(
    TABLE Product
);

CREATE SYNCHRONIZATION USER store31;

CREATE SYNCHRONIZATION SUBSCRIPTION TO "Product"
    FOR store31
    TYPE 'tcpip'
ADDRESS 'host=localhost;port=2439'
    OPTION ScriptVersion ='v1.0';
```

6. リモート・データベースにデータを再ロードします。

```
dbisql -c "dsn=newsales" read newsales_data.sql
```

サブスクリプションを前の手順で定義し、前のデータベースからデータをアンロードするときに外部の再ロードを指定しているため、すべてのデータは、まだ同期されていない挿入としてマーク付けされます。

7. *dbmlsync* を実行し、変更を統合データベースに送信します。再構築用の同期スクリプトとして他のバージョン (Script Version = REBUILD など) を使用するように統合データベースを設定している場合は、そのバージョンをコマンド・ラインで指定できます。これにより、同期サブスクリプションで指定したバージョンが無効になります。

```
dbmlsync -v -c "dsn=newsales" -o rem.txt -e "ScriptVersion=REBUILD"
```

8. 手順 7 の *dbmlsync* が終了したら、それ以降の同期では、サブスクリプションのバージョンの同期を使用します。

```
dbmlsync -v -c dsn=newsales -o rem.txt
```

同期が終了した後は、リモート・データベースの Purchase テーブルのすべてのローが統合データベースにアップロードされています。

各リモート・データベースに対して、上記の手順を繰り返すことができます。前の項で説明したように、統合データベース内の既存のローとアップロードされたローのプライマリ・キーが一致する場合の処理が必要かどうかは、統合データベースの同期ロジックによって異なります。各リモート・データベースで上記の手順を実行した後は、統合データベースにデータが移植または再移植されています。

法的注意

iAnywhere、iAnywhere Solutions、Adaptive Server、SQL Anywhereは、iAnywhere Solutions , Inc.またはSybase, Inc.とその系列会社の商標です。その他の商標はすべて各社に帰属します。

Mobile Linkの技術には、Certicom,Inc.より供給を受けたコンポーネントが含まれています。これらのコンポーネントは特許によって保護されています。

本書に記載された情報、助言、推奨、ソフトウェア、文書、データ、サービス、ロゴ、商標、図版、テキスト、写真、およびその他の資料(これらすべてを"資料"と総称する)は、iAnywhere Solutions ,Inc.とその供給元に帰属し、著作権や商標の法律および国際条約によって保護されています。また、これらの資料はいずれも、iAnywhere Solutions ,Inc.とその供給元の知的所有権の対象となるものであり、iAnywhere Solutions ,Inc.とその供給元がこれらの権利のすべてを保有するものとします。

資料のいかなる部分も、iAnywhere Solutionsの知的所有権のライセンスを付与したり、既存のライセンス契約に修正を加えることを認めるものではないものとします。

資料は無保証で提供されるものであり、いかなる保証も行われません。iAnywhere Solutions ,Inc.は、資料に関するすべての陳述と保証を明示的に拒否します。これには、商業性、特定の目的への整合性、非侵害性の黙示的な保証を無制限に含みます。

iAnywhere Solutionsは、資料自体の、または資料が依拠していると思われる内容、結果、正確性、適時性、完全性に関して、いかなる理由であろうと保証や陳述を行いません。iAnywhere Solutionsは、資料が途切れていないこと、誤りがないこと、いかなる欠陥も修正されていることに関して保証や陳述を行いません。ここでは、「iAnywhere Solutions」とは、iAnywhere Solutions, Inc.またはSybase,Inc.とその部門、子会社、継承者、および親会社と、その従業員、パートナー、社長、代理人、および代表者と、さらに資料を提供した第三者の情報元や提供者を表します。

＊本書は、米国iAnywhere Solutions, Inc.が作成・テストしたものを日本語に翻訳したものです。

アイエニウェア・ソリューションズ株式会社
http://www.ianywhere.jp

1023510