SQL Remote での不一致解決シナリオ

概要：この文書では、不一致解決を使用し、Adaptive Server Anywhere のレプリケート時にデータが同期しなくなることを防止する方法について説明します。

問題：

統合データベースとリモート・ユーザで構成されるレプリケーション・システムを想定します。レプリケートは正常に実行されていますが、データが同期していないことがわかったとします。次に例を示します。

1. 両方のデータベースに 'Dave' という値が含まれています。
2. リモート・データベースで 'Dave' という値を更新し、'Craig' に変更します。この変更をレプリケートし、両方のサイトで最終的に 'Craig'.という値が含まれるようにします。
3. dbremote をリモートで実行し、この変更を統合側に送信します。統合側では、まだ dbremote を実行して変更を適用していません。
4. その間に、統合データベースに 'Dave' という値を挿入し、それを 'Mike' に変更します。これがシステムで最後に行われた変更になるため、値は両方のサイトでレプリケートされるものと想定しています。最終的には、両方のデータベースに 'Mike' という値が含まれるはずです。
5. dbremote を統合データベースで実行し、この変更をリモート側に送信します。
6. dbremote をリモート側と統合側でもう一度実行した後に、データベースを確認します。統合データベースには 'Craig'、リモート・データベースには 'Mike' という値が含まれています。2 つのデータベースが同期していません。

解決策：

これは想定された動作です。不一致解決というテクニックを使用しないかぎり、デフォルトではこのような動作になります。

このような動作になるテーブル・スキーマ例を次に示します。

```
create table t1
(
num integer default autoincrement,
name char(30),
last_modified timestamp NULL DEFAULT CURRENT timestamp,
PRIMARY KEY ("num")
);
```

この問題を回避する方法として、次のものがあります。

次のトリガは、NAME カラムが更新されるごとに、LAST_MODIFIED カラムの値を更新します。

```
create trigger newTime before update of name
on t1
referencing old as old_time new as new_time
for each row
begin
message 'Update trigger fired -- updated the last_modified column.';
set new_time."last_modified" = now(*);
end
```

次のような resolve update トリガをデータベースに追加することで、不一致解決を実行できます。

```
create trigger boom resolve update
on t1
referencing old as old_row new as new_row remote as remote_row
for each row
begin
message 'Resolve update trigger fired!';
if (new_row.last_modified < old_row.last_modified) then
set new_row.name = new_row.name;
end if;
end
```

上記のトリガを追加した場合は、システムの動作が変更されます。不一致が発生すると、データで最後に実行された変更に基づいて、レプリケートすべき値が特定されます。上記のシナリオでは、dbremote をリモート側と統合側でもう一度実行した後は、両方のデータベースに 'Mike' という値が含まれます。

レプリケーション時の不一致解決の詳細については、マニュアルの「Adaptive Server Anywhere の SQL Remote 設計」を参照してください。