Using MobiLink with Oracle Snapshot Isolation

このマニュアルでは、Mobile Link と Oracle 統合データベースでタイムスタンプベースのダウンロードを使用している場合に行う必要がある操作を説明します。

Mobile Link と Oracle 統合データベースでタイムスタンプベースのダウンロードを使用する場合は、すべての変更がダウンロードされていることを保証するための処置を取る必要があるかもしれません。そうしないと、ダウンロード前のコミットされていない変更は、それらの変更がコミットされた後に行われるダウンロードでは欠けている可能性があります。これは、Oracle がスナップショット分離 (非ブロック読み取りとも呼ばれる) を使用するためです。サポートされる他の統合データベースでは、ダウンロードされるローにコミットされていない変更があると、デフォルトでは、その変更がコミットされるまでダウンロードがブロックされます。

Oracle の非ブロック読み取りは、過去の特定の時点に関して最新であるスナップショットとして、トランザクション・ログによって実装されます。長時間にわたるトランザクションのタイムスタンプはコミットされるまで見えないため、長時間にわたるトランザクションを使用すると、時間ベースのダウンロードではデータが欠ける可能性があります。その例を以下に示します。

1. Mobile Link ユーザ ML1 は、時間 T1 までに完全に同期します。
2. 接続 C1 は、長時間にわたるトランザクション TX1 を開始します。
3. 接続 C1 は、時間 T2 (> T1) を使用してタイムスタンプ・カラムの値をロー R1 に挿入するか、または更新します。TX1 は続行されます。
4. ML1 は、時間 T3 (> T2 > T1) までに完全に同期します (または、完全に同期したとみなします)。ダウンロード・トランザクションは R1 を認識できないため、R1 はダウンロードされないことに注意してください。Oracle 以外のデータベースは、この時点でブロックされ、TX1 がコミットされるのを待ちます。
5. 接続 C1 は TX1 をコミットします。

last_download_timestamp は T3 (> T2) であるため、ML1 は R1 を決してダウンロードしません。

最近の EBF で解決された問題

この問題は、最近の EBF で解決されています。SQL Anywhere Studio バージョン 9.0.2.3021 以降、Oracle 統合データベースの場合、Mobile Link サーバは次の last_download_timestamp を、最初に開いたトランザクションの開始時間に設定します。これにより、変更のコミット後に行われる次回の同期ですべての変更が取得されるようになります。Mobile Link サーバによって使用される Oracle アカウントは、V_$TRANSACTION Oracle システム・ビューのパーミッションを持っている必要があります。SYS だけがこのアクセスを許可できることに注意してください。Oracle の構文は以下のとおりです。

grant select on SYS.V_$TRANSACTION to <user_name>

ただし、Oracle でタイムスタンプベースの同期を使用していて、長時間にわたるトランザクションがある場合は、この変更によって、同じ変更が複数回ダウンロードされる可能性が高くなります。これによってデータが欠けることはなく、Mobile Link 同期もそのような重複を効果的に処理しますが、その重複がパフォーマンスに影響する可能性はあります。特に、いくつかのトランザクションが同期間の時間よりも長い場合はその可能性があります。このパフォーマンス問題は、長時間にわたるトランザクションを防止することによって最小限に抑えることができます。

旧バージョンの対処方法

9.0.2.3021 よりも古いバージョンの Mobile Link サーバを使用している場合は、以下の手順によって同等の機能を得ることができます。

トリガを使用して各ローの last_modified マーカを維持している場合、すべての変更がダウンロードされていることを確認する最も簡単な方法は、開いているトランザクションの最も早い開始時間になるように次の last_download_timestamp を変更することです。Mobile Link は、次の last_download_timestamp を変更するための modify_next_last_download_timestamp 接続イベントを提供しますが、そのイベントはダウンロード・トランザクションの終了時に呼び出されるため、ユーザはダウンロード・トランザクションの開始時に新しい値を決定する必要があります。したがって、ユーザは、begin_download 接続イベントで値を決定し、それを modify_next_last_download_timestamp イベントで使用する必要があります。この場合、値はイベント間のパッケージ変数に保存されます。詳しい手順を以下に示します。

1. V$TRANSACTION Oracle システム・ビューにアクセスするためのパーミッションが Mobile Link 接続 (つまり、Oracle ユーザ) にあることを確認します。SYS だけがこのアクセスを許可できます。また、このアクセスは、ベース V_$TRANSACTION ビューに対して許可する必要があります (V$TRANSACTION は、V_$TRANSACTION ビューの同義語です。したがって、V$TRANSACTION にパーミッションを直接付与することはできません)。

```
grant select on SYS.V_$TRANSACTION to <user_name>
```

ストアド・プロシージャにはロール権限がないため、ロールを通じてではなく直接的にパーミッションを持つことはユーザにとって極めて重要です。

ビュー全体へのアクセスを許可したくない場合は、以下の例のように、必要な start_time カラムへのアクセスだけを可能にするビューを代わりに定義できます。

```
create view transaction_start_times
as
select start_time from v$transaction;
```

```
grant select on transaction_start_times to public;
```

2. タイムスタンプ値を保持する変数を備えたパッケージと (Oracle 8 の場合は、TIMESTAMP 型の代わりに DATE 型を使用する)、その変数を設定して (Mobile Link イベントから) 取得するためのプロシージャを作成します。

```
create or replace package SyncVars
as
    procedure SetDownloadTimestamp;
    function GetDownloadTimestamp return timestamp;
end SyncVars;
/

create or replace package body SyncVars
as
    DownloadTimestamp timestamp;

    procedure SetDownloadTimestamp
    as
    begin
    select nvl( min( to_timestamp( start_time, 'mm/dd/rr hh24:mi:ss' ) ),
localtimestamp )
    into DownloadTimestamp
    from v$transaction;
    end SetDownloadTimestamp;

    function GetDownloadTimestamp return timestamp
    as
    begin
    return DownloadTimestamp;
    end GetDownloadTimestamp;

end SyncVars;
/
```

3. タイムスタンプ値を決定するための begin_download 接続スクリプトを追加します。

```
exec ml_add_connection_script( 'version?', 'begin_download',
'SQL','begin SyncVars.SetDownloadTimestamp(); end;' );
```

4. Mobile Link によって使用されるタイムスタンプを置換するための modify_next_last_download_timestamp 接続スクリプトを追加します。

```
exec ml_add_connection_script(
'version?','modify_next_last_download_timestamp', 'SQL','begin ? :=
GetDownloadTimestamp(); end;' );
```

これにより、Mobile Link が疑問符の代わりに渡す値は前の同期で決定されたタイムスタンプ値になるため、テーブルの download_cursor スクリプトでは通常の 'WHERE last_modified >= ?' 句を使用するだけで済みます。

その他のオプション

最初に開いたトランザクションの開始時間を使用したくない場合は、変更されたローだけをダウンロードするときにローが欠けていないことを別の方法で確認できます。最初のオプションは、上記例を少し修正したもので、同期したテーブルが関係するトランザクションだけを考慮します。2 番目のオプションでは、最後のダウンロード時間ではなくローのタイムスタンプを変更します。3 番目のオプションでは、タイムスタンプの使用を完全に回避します。

長いトランザクションのテーブルが同期していない場合

長いトランザクションを備えたテーブルが同期に関係していない場合は、同期したテーブルに保留中の変更がある開いたトランザクションだけを考慮することによって、上記の重複ウィンドウを減らすことができます。対応する SQL を以下に示します (同期したテーブルは所有者が OWNERNAME の TABLE1 と TABLE2 であると仮定しています)。

```
select nvl( min( to_timestamp( start_time, 'mm/dd/rr hh24:mi:ss' ) ),
localtimestamp )
   from v$transaction
   where addr in
      (select taddr
         from v$session
         where sid in
            (select sid
               from v$lock
               where type = 'TM'
```

```
        and id1 in
        (select object_id
          from dba_objects
          where object_name in ('TABLE1', 'TABLE2' )
          and owner = 'OWNERNAME')
    )
)
```

Mobile Link 接続には、ベース・ビューの V_$SESSION、V_$LOCK、および V_$TRANSACTION に対するパーミッションが必要です。

トランザクションが実行間隔の期間よりも長い場合

次の last_download_timestamp を変更するのではなく、COMMIT 後に last_modified タイムスタンプを更新します。

1. last_modified タイムスタンプではなく変更済みフラグをトリガで設定します。
2. last_modified タイムスタンプを定期的に更新して、変更済みフラグが設定されているローに対して変更済みフラグをクリアするバックグラウンド・プロセスを用意します。
3. ダウンロード・カーソルは、変更済みフラグと last_modified タイムスタンプによって選択する必要があります。

これに対する重複ウィンドウは、バックグラウンド・プロセスの実行間隔の期間です。長いトランザクションがバックグラウンド・プロセスの実行間隔の期間よりも長くなることが予想される場合は、この手順が前の手順よりも好ましいやり方です。

Oracle 10g を使用している場合

last_modified タイムスタンプの代わりに、Oracle 10g の組み込み ORA_ROWSCN ロー・マーカを使用することもできます。

ORA_ROWSCN は、ロー (ROWDEPENDENCIES を有効にしてテーブルが作成された場合) またはブロックのコミット・システム変更番号 (SCN) を保持する疑似カラムです。コミット SCN は、トランザクションごとに一意です。

1. ROWDEPENDENCIES を使用してテーブルを作成します。これによってローごとに 6 バイト加算されることに注意してください。
2. last_download_scn を (dbms_flashback.get_system_change_number() を使用して) 確認し、保持します。
3. ダウンロード・カーソルは、ORA_ROWSCN > last_download_scn であるローを選択する必要があります。

この手法には重複や欠けているローはなく、トリガも必要ありません。ただし、この手法では、Oracle 10g、ROWDEPENDENCIES

を使用して作成されたテーブル、およびユーザによる last_download_scn の管理が必要になります。