AL-036 バッチ更新最適化機能

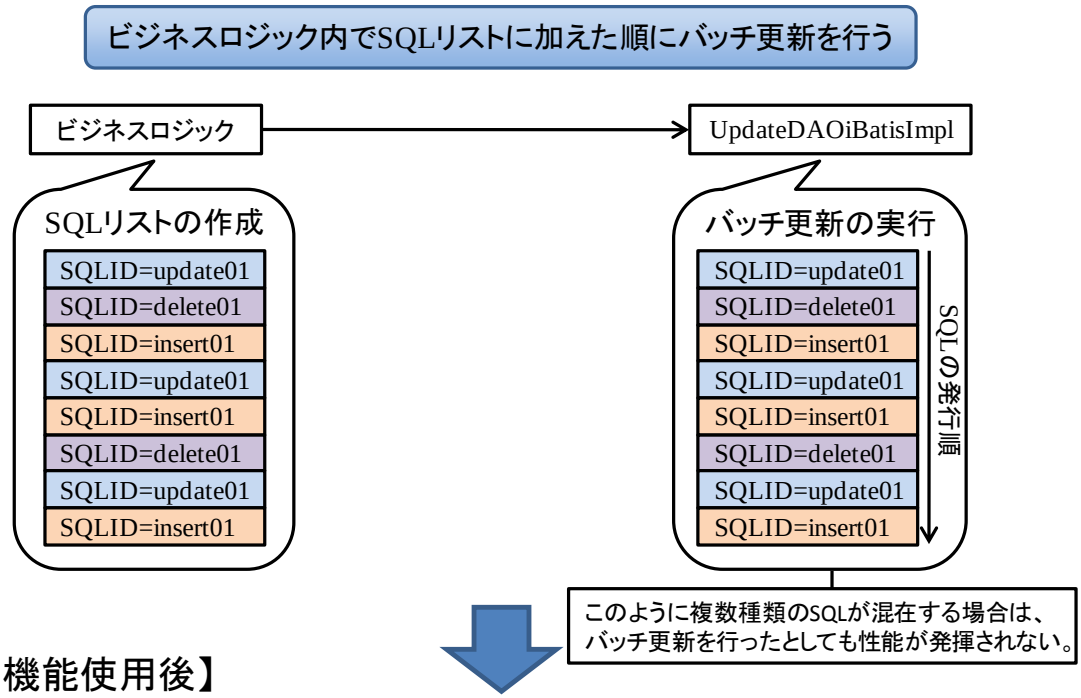
■ 概要

◆ 機能概要

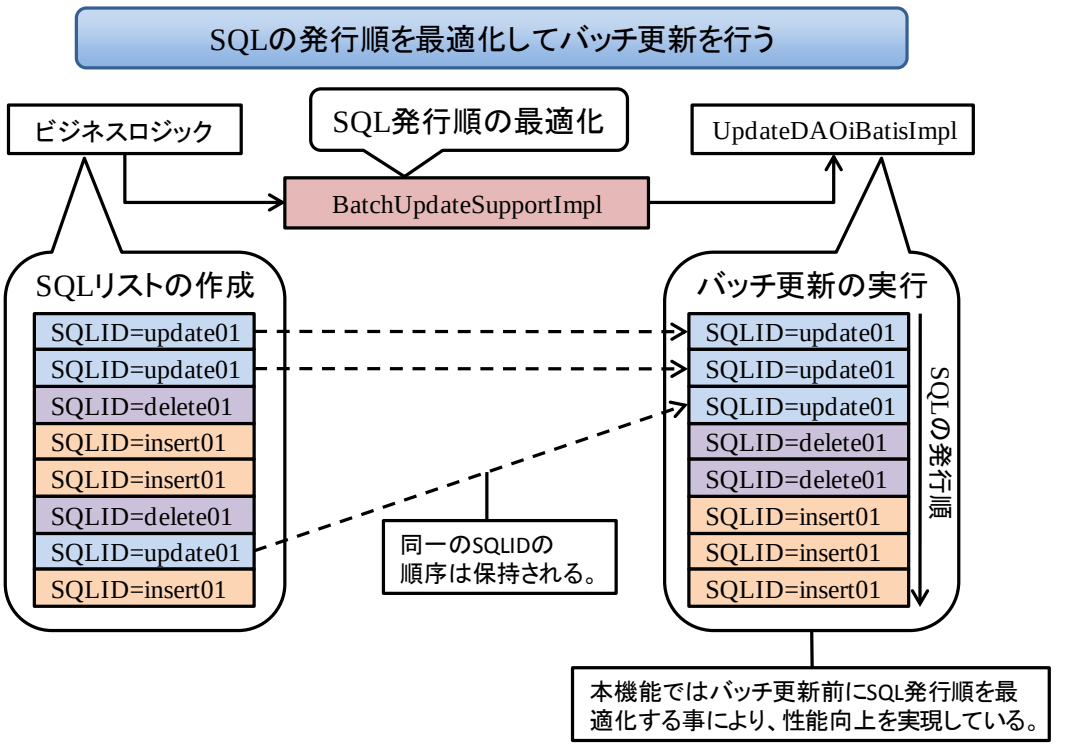
- バッチ更新を行う前に SQL の発行順を最適化する機能を提供する。
- SQL の最適化を行うと、バッチ更新時に発行する SQL の種類が 2 種類以上の場合に、本機能による性能の向上が期待できる。
  - （発行する SQL が単一の場合は、従来のバッチ更新との差は無い）
- 最適化により SQL の発行順が変更される為、「AL-036 バッチ更新最適化機能」を使用する際は、SQL の発行順が変更されても問題が無い事が前提条件となる。

◆ 概念図

【機能使用前】



【機能使用後】



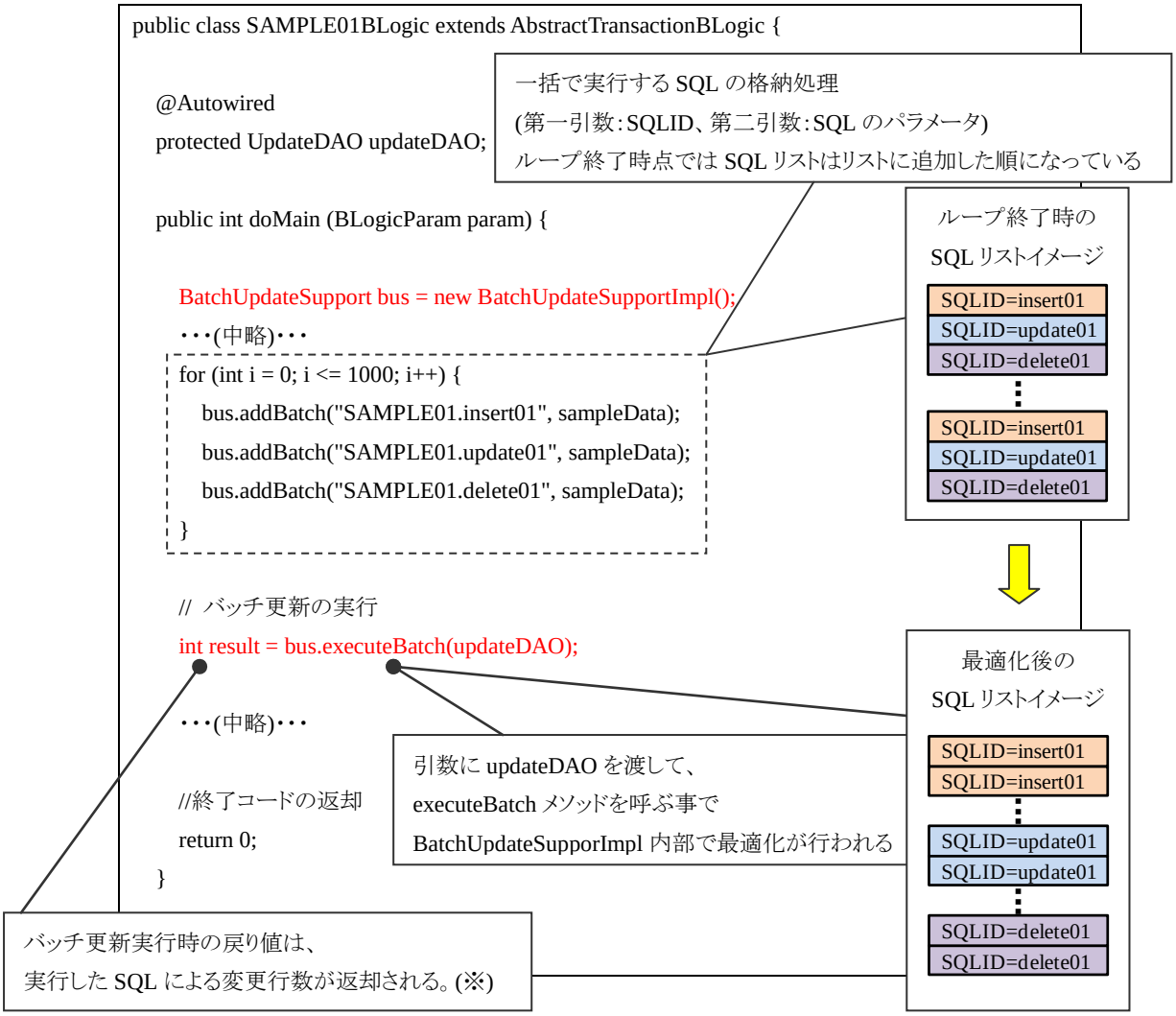
◆ 解説

- SQL 発行順の最適化のために、SQLID をキーにして並び変える。
  - SQL の発行順は、デフォルトではユニークな SQLID が SQL リストに登録された順を保持する。
  - 例えば「A,B,C,D」という４種の SQLID が「C,A,B,D,C,B,A」の順に SQL リストに格納されていたとすると、ユニークな SQLID の順は「C,A,B,D」となる。
    この順番を保持したまま最適化が行われるため、バッチ更新時の SQL 発行順は「C,C,A,A,B,B,D」の順となる。
  - ソート順は後述の方法により変更可能である(後述の sort メソッドを使用するか、拡張ポイントの項目を参照する事)
- 同一の SQLID 間では、概念図中の破線のように、最適化時に SQL リストに格納された順序を保持する。
- 最適化により、同一 SQL を連続して発行する事により、PreparedStatement オブジェクトを有効利用する事が出来る。
  その結果、PreparedStatement オブジェクトの生成回数の減少、通信回数の削減(※)につながり、性能の向上が期待できる。

  ※ PostgreSQL,OracleDatabase については通信回数の削減を確認している。

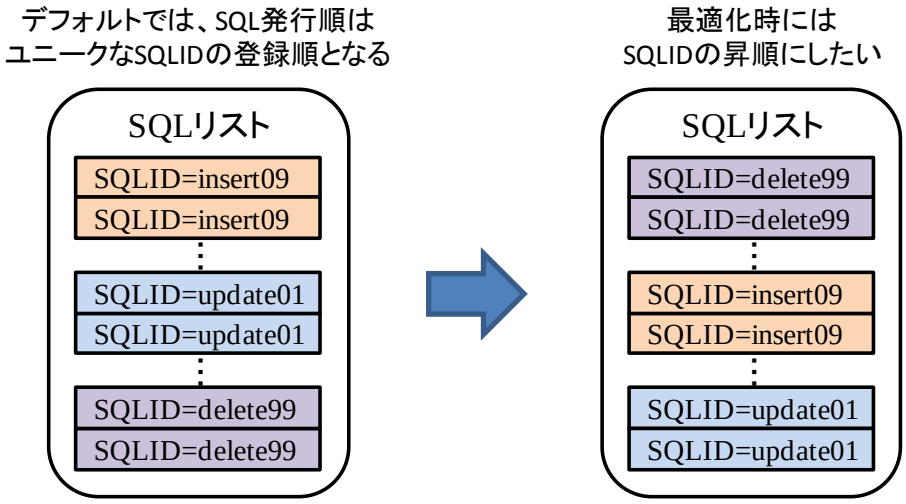
◆ コーディングポイント

- 本機能を使用する際の実装例
  - ビジネスロジック実装例(TERASOLUNA Batch Framework for Java ver 3.x の場合)



    - ※ java.sql.PreparedStatement を使用しているため、ドライバにより正確な行数が取得できないケースがある。
      変更行数が正確に取得できないドライバを使用する場合や、変更行数がトランザクションに影響を与える業務(変更行数が 0 件の場合エラー処理をするケース等)では、バッチ更新は使用しないこと。
      (参考資料)
      http://otndnld.oracle.co.jp/document/products/oracle10g/101/doc_v5/java.101/B13514-02.pdf
      450 ページ「標準バッチ処理の Oracle 実装の更新件数」を参照のこと。
- SQL の発行順序に関する注意点
  - 最適化により、SQL の発行順が変更されてしまうため、SQL の発行順に意味があるような場合は、発行順が保持されるように注意する事。

- sort メソッドを使用して昇順に最適化を行う
  - BatchUpdateSupportImpl クラスが持つ sort メソッドを使用する事により、最適化後の SQL 発行順を昇順に並び変える事が出来る。
  - ビジネスロジック中での sort メソッドの使用例を以下に例を挙げて掲載する。



  - ビジネスロジッククラスの実装例(部分的に抜粋)

```
・・・
BatchUpdateSupport bus = new BatchUpdateSupportImpl();
・・・(中略) ・・・
// SQL 発行順のソート
bus.sort()
// バッチ更新の実行
bus.executeBatch(updateDAO);
・・・
```

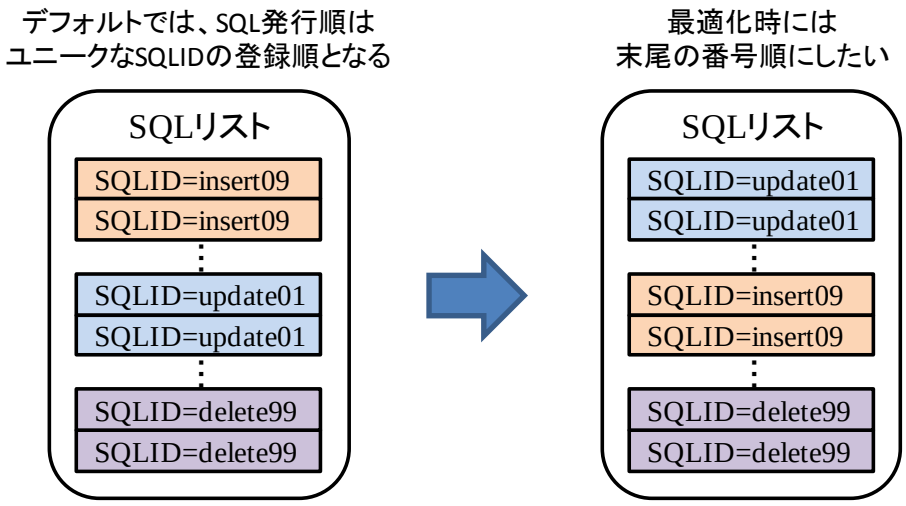
バッチ更新実行前に、sort メソッドを呼び出す事によって、内部的にソートフラグが立つ。

実際にソートが行われるのはバッチ更新実行時。

  - このように実装する事により、バッチ更新実行時に SQL の発行順が SQLID の昇順となるようにソートされる。
  - また、sort メソッドは引数として java.util.Comparator インタフェースの実装クラスを渡す事により、昇順以外の順に並び変える事も可能である。
    （詳細は拡張ポイントの項目を参照する事）

◆ 拡張ポイント

- SQL 最適化時のソート順を変更する方法（sort メソッドを使用する方法）
  java.util.Comparator インタフェースの実装クラスを作成し、バッチ更新実行前に sort メソッドを呼び出す事によって、ソート順を変更する事ができる。
  - ➢ 以下に例を挙げて「java.util.Comparator インタフェースの実装クラス」とビジネスロジックの実装例を掲載する。



➢ java.util.Comparator インタフェースの実装クラスの実装例

```
public class CustomSort implements Comparator<String> {

    public int compare(String str1, String str2) {
        if (str1 != null && str2 != null) {
            String subStr1 = str1.substring(str1.length() - 2);
            String subStr2 = str2.substring(str1.length() - 2);
            return subStr1.compareTo(subStr2);
        }
        return 0;
    }

}
```

Comparator インタフェースの実装
Compare メソッドを作成する。

～このロジックの簡単な説明～
渡された文字列の末尾 2 文字を取得し、比較して昇順になるようにしている。
引数のどちらかが null の場合は 0 を返す。

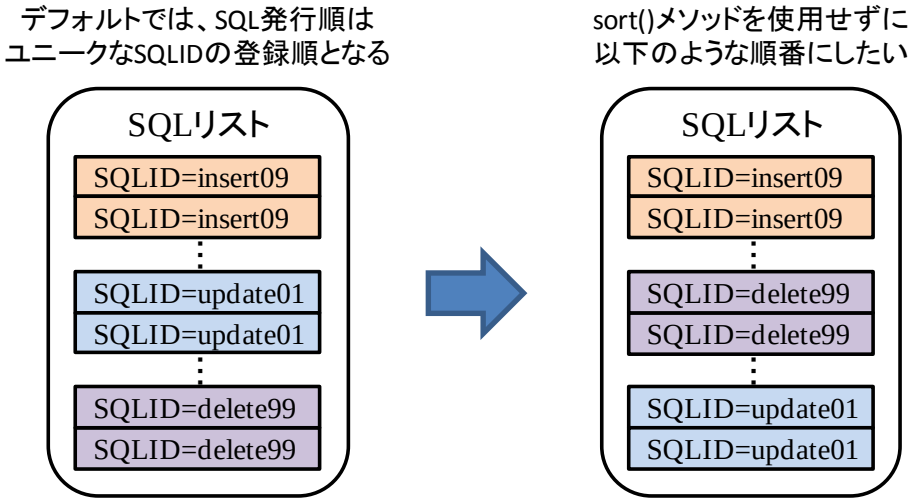
➢ ビジネスロジッククラスの実装例(部分的に抜粋)

```
・・・
BatchUpdateSupport bus = new BatchUpdateSupportImpl();
・・・(中略) ・・・
// SQL 発行順のソート
bus.sort(new CustomSort())
// バッチ更新の実行
bus.executeBatch(updateDAO);
・・・
```

作成した java.util.Comparator インタフェース実装クラスのインスタンスを生成し、sort メソッドの引数として渡す。

実際にソートが行われるのは
バッチ更新実行時。

- SQL 最適化時のソート順を変更する方法（直接 SQLID を指定する方法）
  バッチ更新時に SQLID を直接指定する方法でも、SQL の発行順を変更する事が可能である。
  - 以下に例を挙げて SQLID を直接指定して、SQL の発行順を変更する際のビジネスロジックの実装例を掲載する。



  - ビジネスロジッククラスの実装例(部分的に抜粋)

```
・・・
BatchUpdateSupport bus = new BatchUpdateSupportImpl();
・・・(中略) ・・・
// バッチ更新の実行
bus.executeBatch(updateDAO,
      "Common.insertData09",
      "Common.deleteData99",
      "Common.updateData01");
・・・
```

バッチ更新実行時に、SQLID を文字列で直接指定する。

  - 直接 SQLID を指定する方法を使用する場合は、ビジネスロジック中で SQL リストに格納される可能性のある全ての SQLID を指定しておく必要がある。
  - 以下のようにバッチ更新実行時に、リストに格納されている SQLID の指定がされていなかった場合は、バッチ更新実行時にエラーコード-200 が返却される。
    - ビジネスロジッククラスの実装例(部分的に抜粋)

```
・・・
BatchUpdateSupport bus = new BatchUpdateSupportImpl();
・・・(中略) ・・・
// バッチ更新の実行
bus.executeBatch(updateDAO,
      "Common.insertData09",
      "Common.deleteData99");
・・・
```

SQL リストに格納されている
”Commom.updateData01”を指定していないので
バッチ実行時にエラーコード-200 が返却される。

■ リファレンス

◆ 構成クラス

	クラス名	概要
1	jp.terasoluna.fw.batch.dao.support.BatchUpdateExecutor	バッチ更新一括実行クラス。 オブジェクト内に含まれる複数のバッチ更新を一括で実行する。
2	jp.terasoluna.fw.batch.dao.support.BatchUpdateResult	バッチ更新実行結果クラス。 BatchUpdateExecutor によるバッチ更新一括実行の結果として各々のバッチ更新の結果を、リストにまとめて返却される。
3	jp.terasoluna.fw.batch.dao.support.BatchUpdateSupport	バッチ更新サポートインタフェース。 バッチ更新用の SQL 追加メソッドやバッチ更新実行メソッド等を定義している。
4	jp.terasoluna.fw.batch.dao.support.BatchUpdateSupportImpl	バッチ更新サポートインタフェースの実装クラス。 SQL の最適化やバッチ更新を行う。

◆ 関連機能

- 『BB-01 データベースアクセス機能』

◆ 使用例

- **機能網羅サンプル**(terasoluna-batch-functionsample)

◆ 備考

- 動的 SQL の使用について
  - 本機能は SQLID の並べ替えによる最適化を行うものである。
  - 動的 SQL を使用した場合、同一の SQLID であっても渡されるパラメータにより、毎回異なる SQL が発行される可能性がある。
  - そのため、動的 SQL を使用した場合は静的 SQL を使用する場合と比べ、使用した動的 SQL の複雑さに因って性能向上効果は薄れてしまうと見込まれる。