様　式　Ｃ－１９、Ｆ－１９、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成 ２７ 年　 ６ 月 ２２ 日現在

機関番号：３３８０３
研究種目：基盤研究(C)
研究期間：2012～2014
課題番号：２４５００１３２

研究課題名（和文）オンライン処理中のデータベース一括更新方式の研究

研究課題名（英文）Study of database lump-sum update method during online processing

研究代表者
工藤　司（Kudo, Tsukasa）

静岡理工科大学・総合情報学部・教授

研究者番号：９０５８３７８２

交付決定額（研究期間全体）：（直接経費）　　4,000,000円

研究成果の概要（和文）：データベースの大量のデータに対する一括更新は、オンライン端末からの入力と同時に行うとトランザクションとして実行できないという課題があった。本研究では、ある事実がデータベース内に存在した時間の概念を未来に拡張して、オンライン端末からの入力は現在時刻で更新し、一括更新では過去の時刻のデータを検索して未来の時刻に保存することで両者の競合が回避する方法を提案した。さらに、プロトタイプを構築して評価し、本方式の有効性を確認した。

研究成果の概要（英文）：There was the problem that the lump-sum update to a large amount of data in the database could not be executed as a transaction in the case where the entries from online terminals were executed concurrently with it. In this study, we proposed an update method: it utilized the concept of the time that the fact had existed in the database, and we expanded the time into the future for this method. And, based on this concept, the update from the online terminal is executed to the data of the current time; the lump-sum update queries the data of the past time and stores its update results into the future time. Therefore, the conflicts between the both can be avoided. Furthermore, we composed the prototypes implementing this method and confirmed the validity of it by the experiments.

研究分野：経営情報システム

キーワード：データベース　トランザクション　同時実行制御　バッチ処理

１．研究開始当初の背景
(1) 電子商取引、電子申請などインターネットビジネスの拡大に伴い、不特定多数のユーザに対するノンストップ・サービスが急速に普及している。

(2) 一方、業務システムは多数のオンライン端末からのトランザクション処理と、大量の一括更新を行うバッチ処理から構成される。

(3) このため、両者を安全に同時実行する方式が実用化されているが、従来の方式では以下のような課題があった。
①バッチ処理で長時間に渡ってデータを占有すると、オンライン端末に長時間の待ちを発生させる。
②これを防ぐため、バッチ処理を細分化して実行すると、バッチ処理全体としてトランザクションの ACID 特性が維持されない。

２．研究の目的
(1) ある事実がデータベース内に存在した時間であるトランザクション時間を未来に拡張し、オンライン端末からの入力は現在時刻に保存し、バッチ処理による更新では過去の時刻のデータを検索し、未来の時刻に保存することで両者の競合が回避できることを着想した。

(2) そこで、この着想に基づく更新方式を提案し、両者を同時に実行した場合であっても、バッチ処理による大量の更新をトランザクションの ACID 特性を維持しながら実行でき、かつオンライン端末に長時間の待ちを発生させないことを示す。

３．研究の方法
(1) 業務システムにおけるバッチ処理での更新を体系化し、各々の更新に提案した方式を適用した場合であっても、バッチ処理がトランザクションとして実行され、オンライン端末に長時間の待ちが発生しないアルゴリズムを研究する。

(2) 上記の方式の機能をプログラムとして提供するために、使用しやすい構造を持たせた実装方法を研究、評価する。

(3) 実証研究として具体的な業務システムを想定したプロトタイプを構築し、提案方式の適用評価を行う。さらに従来の更新方式と比較した効率性や、更新時に障害が発生した場合の安全性の評価を行う。

４．研究成果
(1) 提案した方式のアルゴリズムに関する検討を行い、理論的な側面からバッチ処理の更新をオンライン端末からの更新と分離して実行できることを示すと共に、リレーショナルデータベースの SQL で実装し、シミュレーションにより所定の結果が得られることを示した。本成果は、５章〔学会発表〕⑩の国際会議で発表した。本方式の概要を図１に示す。本方式では、トランザクション時間を活用し以下のアルゴリズムで更新を行う。
① バッチ処理の更新は図の①「バッチ更新」に示すように過去の時刻のデータを使用して未来に対して更新する。
② 一方で、オンライン端末からの入力は図の②「オンライン入力」に示すように現在時刻で更新する。
③ ただし、バッチ処理中に行われたオンライン端末からの入力結果に対してもバッチ処理を行う必要があるため、これをバッチ処理と同様に図の(3)「OB 更新」（オンラインーバッチ更新）として未来に保存する。
④ バッチ処理で設定された未来の時刻が来ると、上記の①から③の全てのデータがデータベースに保存された状態になる。そこで、図の④に示すように全てのデータの中から有効なデータを検索することで、有効な更新結果が得られる。この場合には、③の「OB 更新」がオンライン端末からの入力と、バッチ処理の両方の更新が反映されており、有効なデータになる。

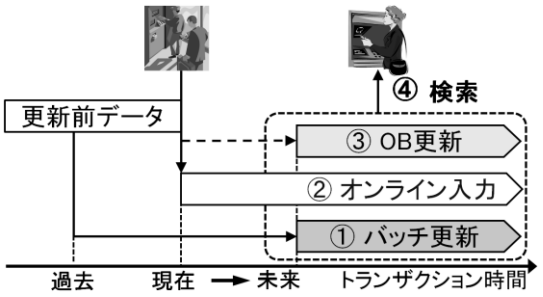

図１　提案する更新方式の概要

(2) バッチ処理における更新処理を調査・体系化し、分離データモデル、個別更新データモデル、相互関連データモデルの３つのパターンに区分できることを示した。さらに、自治体で広い地域の住所の表記を一括して変更する住居表示業務を取り上げ、このような更新処理を行う場合には、従来の更新方式ではバッチ処理をトランザクションとして処理できないことを示した。すなわち、住所は世帯単位に決まる。一方で、住民は個人単位で転居や転入、転出などを行うため、更新の単位は個人になる。従って、世帯単位の住所の更新をバッチ処理で実行しながら、個人単位の転居などの異動をオンライン端末から実行すると矛盾が発生することを示した。さらに、本方式を Java プログラムで実装し、オンライン端末からの入力を待たせることなくバッチ処理を分離した処理として実行できることを示した。本成果は、５章〔雑誌論文〕②に掲載された。

(3) 本方式の応用分野を検討する中で、分散

データベース環境における一括更新が効率的に行えることが分かってきた。これは、従来の方式で大量のデータに対する一括更新を、オンライン端末からの入力の遅延を小さくしたままで実行するためには、更新を分割して実行する必要があることに起因する。更新を分割することで、データ更新の確定（コミット）の操作が更新単位に必要になるが、複数のサーバに跨る分散データベースでは、両方のサーバの同期を取るために 2 相コミットが必要になる。これは、分散環境で実施されるため効率性に課題があり、実行回数の増大に伴い大きな効率の低下が発生する。一方で、提案の方式ではコミットは図 1 に「未来」で示される時点で実行するため、複数の更新をまとめて効率良くコミットできる。そこで、従来方式と分散データベース環境で処理効率を比較したところ、図 2 に示すように提案方式では例えば、200 件を一括してコミットした場合には 60 倍の性能を得られた。ここで、従来方式としてはミニバッチを使用している。この成果は、５章の〔学会発表〕⑥に記載した国際会議で発表した。

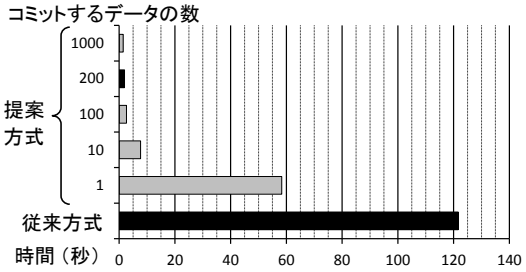

図 2　提案方式と従来方式の効率の比較

(4) 一方で、分散データベース環境ではデータベース更新時間がネットワークやサーバの負荷などの環境要因で変化する。このため、バッチ処理の完了時間を長く設定する必要があり、図 1 の「OB 更新」実行に伴う不要な負荷が発生することが分かった。そこで、図 2 に示すように、データテーブルには先頭が「@」の仮の時刻を設定しておき、この仮テーブルとバッチ処理の終了時刻を「対応テーブル」に保存する方法を考案した。ユーザにはビュー表を提供する。バッチ処理が終了した時点で「対応テーブル」に「終了時刻」を設定することで、データテーブルの仮時刻がビュー表で「終了時刻」に置き換わる。なお、修了前は「対応テーブル」の 2 行目に示すように空値（null）を設定しておく。この方式によって、バッチ処理終了と同時に「OB 更新」も停止し、効率的に終了できることを実験により確認した。本成果は、５章の〔雑誌論文〕①に掲載された。

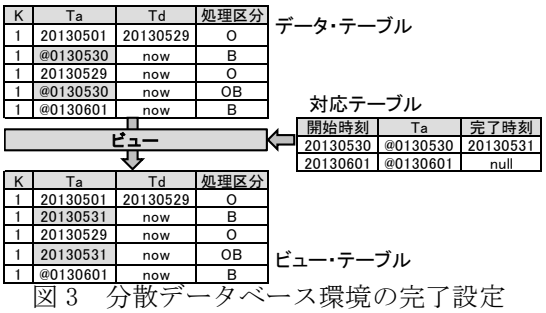

図 3　分散データベース環境の完了設定

(5) バッチ処理をトランザクションとして実行するためには、オンライン端末からの入力との間で直列化可能スケジュールを構成する必要がある。すなわち、バッチ処理がオンライン入力のある更新の直後で、かつ、その次の更新の前に実行されたのと同様の結果を得る必要がある。このため、図 4 に示すようにバッチ処理（図の「バッチ更新」；およびコミットを「c」で表記）とオンライン端末からの入力（図の 0_1、0_2、0_3、...）の間に直列化可能スケジュールを構成するための同時実行制御を業務システムのプロトタイプも実装し、同時実行制御の有効性を実証した。この成果は、５章の〔学会発表〕③の国際会議で発表した。

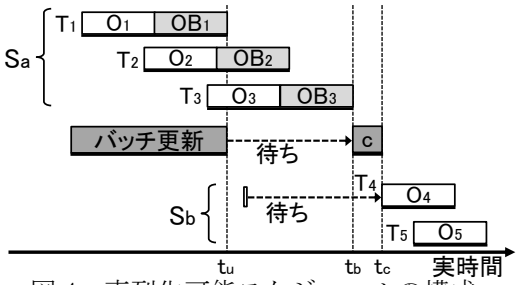

図 4　直列化可能スケジュールの構成

(6) バッチ処理では大量のデータを扱うため、障害が発生した場合には更新前の状態に戻すためのロールバックが必要になる。ここで、実際の運用でもバッチ処理の要件からリソースの不足や、データの品質に起因した障害がしばしば発生する。提案の方式では図 1 に示すように、オンライン端末からの入力とは独立にバッチ処理のデータを保存している。従って、オンライン端末からの入力と同時実行したとしても、これに影響を与えることなくロールバックできることが期待される。そこで、バッチ処理の障害回復機能を実装し、同時に実行されているオンライン端末からの入力に影響を与えることなくロールバックできることを実証した。これにより、実際の運用でもバッチ処理の安全性が高められると考えられる。本成果は、５章の〔学会発表〕④の国際会議で発表した。

(7) 実際の業務システムに適用した場合にもオンライン端末との同時実行がトランザクションとして実現できることを検証するた

め、上記で示した事例以外の業務システムのプロトタイプにも本方式に基づく以下の機能を組込み、実験により所定の効果が得られることを確認した。これらの成果は、５章の〔学会発表〕①、②、⑧、⑨で発表した。
① 輸送運賃計算システムにおいて運賃計算を実行中に、計算の根拠となる単価データを再計算する機能
② 所要量計算システムにおいて、材料費の単価を随時更新しながら、所定の製品の全材料発注額を一括で計算する機能
③ 選挙システムにおいて投票と集計を同時に実行する機能
④ 店舗システムにおいて、販売データの入力を行いながら、旧システムから新システムに一括でデータ移行を行う機能

５．主な発表論文等
（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計　２件）
① Tsukasa Kudo et al., Application of a Lump-sum Update Method to Distributed Database, Int. Journal of Informatics Society, 査読有, Vol. 6, No. 1, 2014, 11-18, http://www.infsoc.org/journal/vol06/IJIS_06_1_011-018.pdf

② Tsukasa Kudo et al., Evaluation of Lump-sum Update Methods for Nonstop Service System, Int. Journal of Informatics Society, 査読有, Vol. 5, No. 1, 2013, 21-28, http://www.infsoc.org/journal/vol05/IJIS_05_1_021-028.pdf

〔学会発表〕（計１０件）
① 岩田力、工藤司、人間の経験知を反映した輸送運賃計算システムの構築と評価、情報処理学会第 77 回全国大会、(20150317-20150319)、京大（京都）

② 新井一也、工藤司：ビッグデータを活用した所要量計算システムの提案、情報処理学会第 77 回全国大会、(20150317-20150319)、京大（京都）

③ Tsukasa Kudo, An implementation of concurrency control between batch update and online entries, Knowledge-Based and Intelligent Information & Engineering Systems 18th Annual Conference, (20140915-20140917), Gdynia (ポーランド)

④ Tsukasa Kudo, Implementation and Evaluations of Recovery Method for Batch Update, Proc. of Int. Workshop on Informatics (IWIN2014), (20140910-20140912), Prague(チェコ)

⑤ 工藤司、バッチ更新とオンライン入力の同時実行制御方式～実装と評価～、電子情報通信学会 SWIM 研究会、(20140523)、機械振興会館（東京）

⑥ Tsukasa Kudo, A Mass Data Update Method in Distributed Systems, 17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems - KES2013, (20130909-20130912), 北九州

⑦ Tsukasa Kudo, A Study of Long Time Transaction, Proceedings of Symposium on Advanced Information Systems (SAIS2013), (20130906), Copenhagen(デンマーク)

⑧ 原田健吾、武田由衣、工藤司、履歴に基づく当選確率を採用した即時抽選システムの提案、情報処理学会第 75 回全国大会、(20130306-20130308)、東北大（仙台）

⑨ 吉永豊、武田由衣、工藤司、ノンストップサービス・システムにおける移行方式の提案、情報処理学会第 75 回全国大会、(20130306-20130308)、東北大（仙台）

⑩ Tsukasa Kudo, A batch Update Method of Database for Mass Data during Online Entry, Proceedings of 16th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems – KES 2012, (20120910-20120912), San Sebastián(スペイン)

〔その他〕
工藤研究室ホームページ　研究内容
http://www.sist.ac.jp/~kudo/research.html

６．研究組織
(1)研究代表者
工藤　司（KUDO, Tsukasa）
静岡理工科大学・総合情報学部・教授
研究者番号：　90583782

(2)研究協力者
岩田　力（IWATA, Chikara）
新井　一也（ARAI, Kazuya）
原田　健吾（HARADA, Kengo）
吉永　豊（YOSHINAGA, Yutaka）