入力中のデータベースに対する訂正機能を持つ一括検索方式

A Database Query Method with Correction for Batch Processing during Online Entry

工藤　司*，武田　由衣**，石野　正彦***，五月女　健治****，片岡　信弘*****
Tsukasa KUDO, Yui TAKEDA, Masahiko ISHINO, Kenji SAOTOME and Nobuhiro KATAOKA

Abstract: In many mission-critical systems, online entry data are accumulated in their databases and regularly processed by batch processing. So, for the efficient system operations, the both have to be executed concurrently. And, previous studies have proposed the various kinds of methods to maintain the integrity of the query result in batch processing even during online entry. However, in the actual system operations, there is incorrect entry data that takes time by being entered. And, it brings up the problem that the integrity of query results can't be maintained by the conventional query methods. This paper proposes a query method, by which the snapshot of the designated time reflects the corrections entered after the time. And, we show that the query result, which has integrity and reflects the correction of error data, can be queried by this method, even while online entry. Moreover, we applied it to an actual mission-critical system and confirmed its effect.

1．はじめに

小売，金融，製造などの基幹系システムでは，多数のオンライン端末から同時にデータが入力（以下，オンライン入力と略記）されるため，トランザクション処理による同時実行制御がおこなわれる[1]．一方で，入力されたデータの定期的な集計などはバッチ処理[1]で実行される．例えば，小売システムでは店舗の販売情報はトランザクション処理により即時にデータベースに反映され，一方で日々あるいは定期の決算はバッチ処理によって行われる．ここで，バッチ処理は大量のデータを一括して処理するために長時間データベースを占有する必要があり，オンライン入力の時間帯を避けて夜間に行われてきた．しかし，近年ではインターネットビジネスの進展などによりオンライン入力時間帯が延長されバッチ処理時間の制限が必要になっている．このため，バッチ処理が所定の時間内に完了しないという，いわゆる「突き抜け」が問題になってきた[2]．

これに対し，オンライン入力中にも，バッチ処理により一貫性のあるデータベース検索結果を得るための方式が実用化されている．たとえば，特定時刻現在のデータベースの状態を検索する場合，データベースの版管理を行うマルチバージョン同時実行制御が広く使用されている[3]．また，筆者らは時間の経過に伴うデータの履歴を管理する時制データベース[4-6]のうち，データが実世界で有効だった有効時間と，データベース内で有効だったトランザクション時間を管理するバイテンポラルデータベース[7,8]により，実世界の特定時間のデータを検索する際に，データの誤り訂正を反映した結果を検索できることを示した[9]．

ところが，決算などの処理では指定された締切り時刻までにオンライン入力されたデータ，すなわち特定時刻現在のデータベースの状態を対象とし，誤りがあればこれを訂正した上で処理が行われる．この場合には，特定の時刻現在のデータに，それ以降の訂正を反映する必要があるためマルチバージョン同時実行制御では対応できない．さらに，バイテンポラルデータベースでも，実世界の状態がデータベースに反映されるまでに時間を要するシステムでは対応できないという課題がある．

本研究では，このような場合でもオンライン入力と並行してバッチ処理を実行するための方式として，訂正検索方式を提案する．本方式では，トランザクション時間を管理するトランザクション時間データベース[4]において，締切り時刻現在の状態に，検索を行う時刻までに発生した訂正を反映する．これにより，オンライン入力の影響を受けずに，訂正を反映したデータベースの状態を検索できることを示す．さらに，これを基幹系システムに実装し，夜間のバッチ処理を削減できるというシステム運用上の効果を確認した．

本論文の構成は以下の通りである．2節で本論文の解決しようとする課題を述べ，3節で課題を解決するための検索方式を示す．4節で基幹系システムにおける適用事例を述べ，5節で適用結果を評価し，6節で考察する．

2．バッチ処理におけるデータベース検索上の課題

2.1　基幹系システムにおけるバッチ処理の構成

基幹系システムでは，データベースにおけるトランザクション処理の一貫性制御や，業務プログラムによるデータのチェック機能により，オンライン入力されたデータに対する一貫性制約が維持される．しかし，大量のデータ検索

2012年3月2日受理
*　総合情報学部　人間情報デザイン学科
**　三菱電機インフォメーションシステムズ株式会社
***　福井工業大学　経営情報学科
****　法政大学大学院　情報科学研究科
*****　東海大学　情報通信学部

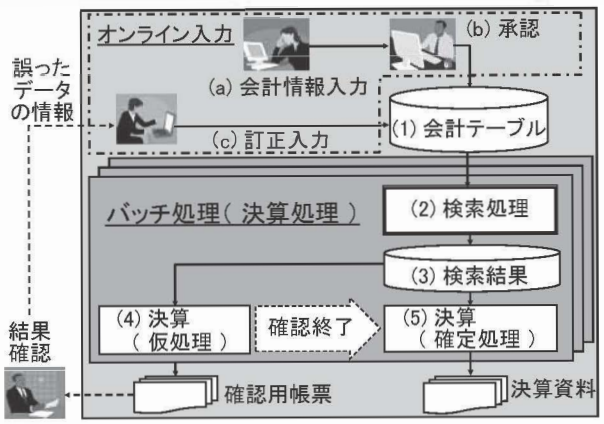

図1 バッチ処理の構成事例

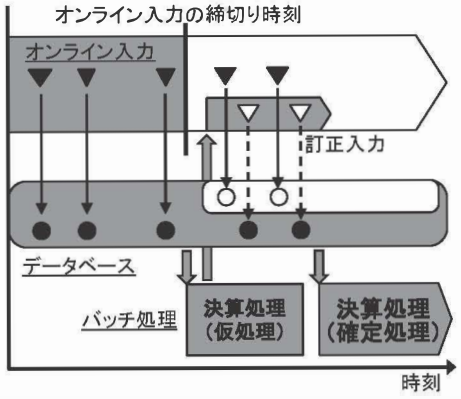

図2 決算における会計データの時系列の推移

を必要とする確認処理，例えば，実際の現金や現品と照合するための集計処理や，複数テーブル間の整合性の確認などをオンライン入力の都度行うことは効率上の課題がある．すなわち，データベースが対象とする実世界の状態を正しく反映している，という定義[10-11]における一貫性の維持は困難である．従って，バッチ処理では，最初にデータベースの一貫性確認が行われる．

図1に会計システムにおける決算処理のデータフローの例を示す．会計情報入力(a) で，支払依頼や入金などの様々な会計データがオンライン入力され，承認(b) を経て会計テーブル(1) に蓄積される．これらの入力・承認は常時行われるため，月次，期，年度の決算処理では締切り時刻が設定され，検索処理締切り時刻までに入力されたデータを対象にして検索処理(2)が行われて検索結果(3)が抽出される。次いで，検索結果に対して仮の決算処理(4)で確認作業が行われ，データ誤りが検出された場合には訂正入力(c) が行われて，処理(2)，(4) が再実行される。こうして確認が完了した後に，確定処理(5) により決算資料の作成が行われる。

ここで，このバッチ処理をオンライン入力と並行して実施するためには，図1のバッチ処理における検索処理（以下，バッチ検索処理と記載）をオンライン入力の影響を受けずに実行できることが必要になる．すなわち，訂正入力

(1) 即時入力の場合の検索対象データ

ID	Va	Vd	Ta	Td	金額	検索対象
001	4/19	4/20	4/20	4/21	1,000	
001	4/19	4/20	4/21	now	1,500	●
002	4/20	4/21	4/21	now	3,000	

(2) 入力遅れのある場合の検索対象データ

ID	Va	Vd	Ta	Td	金額	検索対象
001	4/19	4/20	4/20	4/21	1,000	
001	4/19	4/20	4/21	now	1,500	●
002	4/19	4/20	4/21	now	3,000	●

図3 バイテンポラルデータベースにおける訂正の検索

が行われた場合であってもバッチ検索処理において，業務で発生するオンライン入力データと，訂正で発生するオンライン入力データを識別し，締切り時刻以降は訂正データのみを検索対象とする必要がある．

図2に，図1の決算処理におけるデータの状態を時系列で示す．決算処理では図2の締切り時刻までに入力されたデータを対象に，仮処理としてデータの確認が行われ，データ誤りの訂正後に決算処理の確定処理が行われる．ここで，図2で「▼」は通常の業務入力を，「▽」は訂正入力を示す．また，「●」，「○」はデータベースに蓄積されたデータであり，このうち，●が締切り時刻以前の業務入力および，それに対する訂正入力データを示す．締切り時刻以降に入力されたデータについては，締切り時刻前に入力されたデータの訂正入力のみを抽出し，締切り時刻現在のデータベースの状態に反映した検索結果が決算処理の対象となる．すなわち，図2の事例では，データベースのデータのうち，●のデータが確定処理の対象になる．

2.2 従来のデータベース検索方式における課題

従来のマルチバージョン同時実行制御では時刻の推移によってデータベースの版が管理される．従って，データの訂正を伴うバッチ検索処理へ適用した場合には，締切り時刻以降の入力に対して通常の業務入力と訂正入力が区分されない．すなわち，図2において○で示される，締切り時刻以降の業務入力データも検索対象になってしまうという課題がある．

この課題に対し，我々はバイテンポラルデータベースを実際の基幹系システムに適用することにより，業務システムのオンライン入力中であっても，訂正のみを反映したバッチ検索処理が可能であることを示した[9]．バイテンポラルデータベースでは有効時間とトランザクション時間の双方の時間が管理され，一度追加されたデータは履歴として残される．すなわち，データベース内と実世界の双方の履歴が蓄積される[5,6]．これらの時刻は，例えば企業の人事管理システムであれば有効時間は該当職位に在職した時間，トランザクション時間はそれがデータベース内で有効であった時間となる．なお，これらの時間を，いずれも管理しないデータベースは，スナップショットデータベースと呼ばれる[12]．

図3に，会計システムの旅費精算においてバイテンポラ

ルデーベースを適用し，締切り時刻を 4 月 20 日として訂正データを検索した例を示す．図で，$[V_a, V_d)$は有効時間の時区間，すなわち出張期間を，$[T_a, T_d)$はトランザクション時間の時区間，すなわちこの伝票がシステムに存在した時区間を示す．なお，時刻は 1 日単位としている．4 月 20 日にID=001のデータが入力され，4 月 21 日にこのデータの金額訂正と，新たにID=002のデータの入力が行われた．ここで，旅費精算データの集合をD= {d }とし，有効時間t_vとトランザクション時間t_tを指定したとき，

$$D1 = \{d | d \in D \wedge t_v \in [d[V_a], d[V_d]) \wedge t_t \in [d[T_a], d[T_d])\} \quad (1)$$

なるデータが指定した時間のスナップショットとして検索される．ここで，$d[V_a]$，$d[T_a]$は各々データdにおける属性V_a，T_aの値を，「[ )」は時間の半開区間を示す．

従って，図 3 の(1)に示すように，t_v =4 月 19 日，t_t =4 月 21 日を指定することにより，訂正後のID=001のデータが検索され，ID=002のデータは検索対象外となる．ここで，「now」[13, 14]は時間の推移と共に変化する現在時刻を示す．従って，$d[T_d] = now$ の場合，現在時点でデータは有効となる．

ところが，実際の業務では実世界の状態は必ずしも即時にデータベースに反映されるとは限らない．図 3 の(2)に，(1)のID=002の出張の有効時間が$[4/19, 4/20)$であり，旅費精算の入力が遅れて 4 月 21 日に入力された場合を示す．この場合には，式(1)の条件に合致してしまい，締切り時刻以降の業務による入力であるにもかかわらず検索対象になってしまうという課題がある．さらに，対象とする業務によっては有効時間を管理する必要がないものがある．例えば，ペーパーレスの流れの中では会計システムにおける購入や支払いの伝票はシステムで管理され，実世界における伝票，すなわち帳票としての伝票の運用は少なくなる．従って，伝票の有効時間を管理するバイテンポラルデータベースの適用は非効率である，という課題がある．

3. データ訂正を反映した検索方式の提案

本節では，オンライン入力中であっても，指定時刻までに入力されたデータに対する訂正を反映した結果を得るための検索方式として，訂正検索を提案する．

3.1 訂正検索

訂正検索はトランザクション時間を管理するデータベース，すなわちトランザクション時間データベースを対象とする．トランザクション時間データベースのリレーションRは

$$R(K, T, A) \quad (2)$$

で表現される[11]．各々の属性を以下に示す．

・$K = \{K_1, K_2, \ldots, K_m\}$

指定したトランザクション時間のスナップショットにおける主キー属性集合を示す．

・$T = \{T_a, T_d\}$

トランザクション時間の時区間属性で，ユーザには公開されない．T_aは該当データがデータベースに追加された追加時刻，T_dはデータが論理的に削除された削除時刻を示す．データが削除されていない場合にはT_dの属性値はnowで表現される．

・$A = \{A_1, A_2, \ldots, A_m\}$

その他の属性集合を示す．

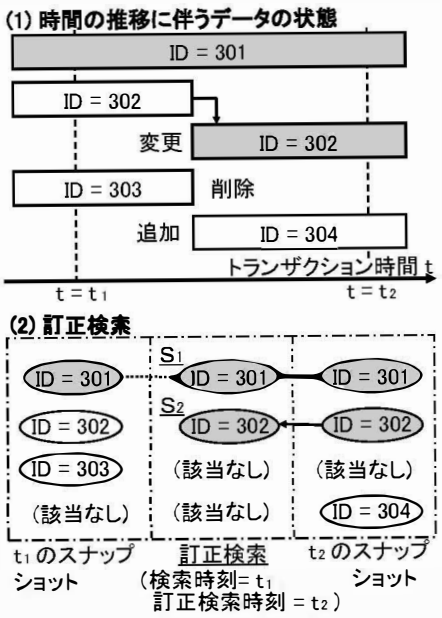

図 4　訂正検索による検索事例

トランザクション時間データベースでは，過去の任意の時刻におけるスナップショット，すなわちデータベースの状態が検索できる．指定時刻をtとしたとき，このスナップショットのデータの集合$Q(t)$は以下の式で表現される．

$$Q(t) = \{q | q \in R \wedge q[T_a] \leq t \wedge t < q[T_d]\} \quad (3)$$

ここで，$q[T_a]$はqの属性T_aの属性値集合を示す．訂正検索は締切り時刻$t = t_1$のスナップショットに加え，データ訂正後の時刻t_2を指定して検索を行い，t_1のスナップショットに対し$t = t_2$までに行われた訂正を反映したものを検索結果とする．以下で，t_1を検索時刻，t_2を訂正検索時刻と呼ぶ．なお，上記の前提から，$t_1 < t_2$となる．

Rの訂正検索のリレーションは，下記のS_1，S_2の和集合$S = S_1 \cup S_2$で表現される．

・S_1 ： t_1からt_2までに変更・削除されていないデータ

S_1に属するデータに対しては，t_1におけるスナップショットのデータを，そのまま訂正検索の対象にする．すなわち，該当データは時刻t_1とt_2のいずれにも存在するため，以下で表現される．

$$S_1 = \{s | s \in Q(t_1) \wedge s \in Q(t_2)\} \quad (4)$$

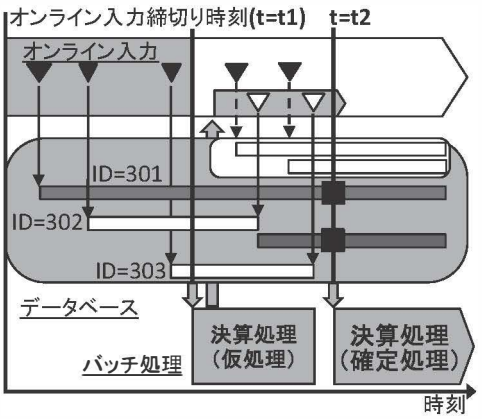

図 5 訂正検索の決算処理への適用事例

・S_2 ：t_1からt_2までに変更されたデータ

変更後のデータを訂正検索の対象にする．変更前後のデータは主キー属性値集合$r[K]$と$s[K]$で対応付けられるため，該当データは式(4)で表現される．この定義からt_2までに削除されたデータは検索対象外になる．

$$S_2 = \{s | s \notin Q(t_1) \wedge s \in Q(t_2) \wedge \exists r \in Q(t_1);\ r[K] = s[K]\} \quad (5)$$

なお，訂正検索結果のデータはスナップショット$Q(t_2)$の部分集合で，通常のトランザクションによる更新結果であり，データの一貫性制約は維持されている．

訂正検索による検索の例を図 4 に示す．図 4 で，トランザクション時間$t = t_1$が検索時刻，$t = t_2$が訂正検索時刻であり， 図の(1)に示すように入力されていたデータ$ID = 301, 302, 303$のうち，t_2までに$ID = 302$が変更，$ID = 303$が削除され，新たに$ID = 304$が追加される．図の(2)にこれらのデータに対する訂正検索の結果を示す．まず，式(4) により$ID = 301$が，また，式(5) により変更後の$ID = 302$が検索され，削除された$ID = 303$，および新たに追加された$ID = 304$は検索対象外となる．

3.2 訂正検索の効果

訂正検索により 2.2 節の課題が解決できることを示す．図 5 に，図 2 に示した決算処理に訂正検索を適用した事例を示す．ここで，データベースのデータは図 4 と同様に時系列での変化を示し，また，オンライン入力の表記▽，▼は図 2 と同様である．締切り時刻前に入力されたデータに対して決算の仮処理が行われ，データ確認の結果$ID = 302$の変更，$ID = 303$の削除の訂正入力が行われる．一方で，締切り時刻以降も業務データのオンライン入力は継続される．

これに対して，締切り時刻$t = t_1$を検索時刻，決算処理開始時刻$t = t_2$を訂正検索時刻として訂正検索を行った場合の検索対象データは，図の「■」時点のデータが該当する．すなわち，締切り時刻以前に入力されたデータは訂正の反映された状態が検索対象となり，締切り時刻以降に入力された業務データは検索対象とはならない．従って，オンライン入力中であっても，オンライン入力の影響を受けずに締切り時刻現在のデータベースに訂正を反映した結果を検索することができる．

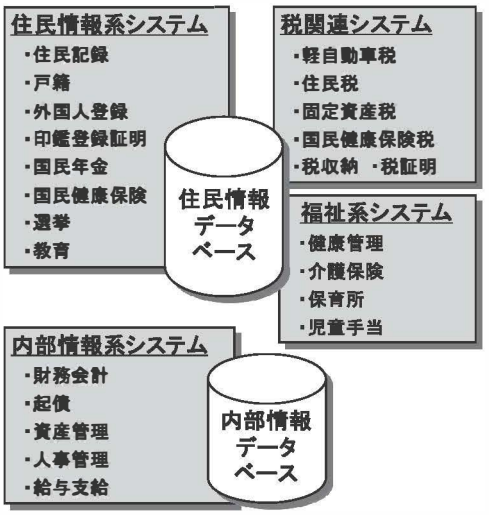

図 6 自治体システムの構成

4. 基幹系システムへの適用

訂正検索を基幹系システムである自治体システムに適用し，実際に運用した結果を示す．

4.1 自治体システムの概要

自治体システムは地方自治体の行政事務処理を支援するシステムであり，図 6 に示す様に各種の業務システムから構成される．大きくは以下の様に区分される．

(a) **住民情報系システム**：住民票，印鑑登録証明などの住民の台帳管理，証明書発行業務

(b) **税関連システム**：自治体管掌地方税の課税，徴収事務および税証明の発行業務

(c) **福祉系システム**：保育所，児童手当などの資格管理，徴収，支給事務などの福祉行政業務

(d) **内部情報系システム**：人事管理，給与支給，財務会計事務などの，自治体内部業務

各業務システムでは届出や申告が窓口で受付けられてオンライン入力され，データベースに蓄積される．また，大量のデータ検索を伴う処理は，定期，あるいは随時のバッチ処理として処理される．

4.2 バッチ処理の運用事例

適用システムにおけるバッチ処理の運用事例として，4.1 節 (b)に示した税関連システムの軽自動車税における修更正を事例として以下に示す．軽自動車税は，住民が所有する軽自動車に対する税であり，住民の申告に基づき 4 月 1 日を基準日として一定期間後に課税が行われる．修

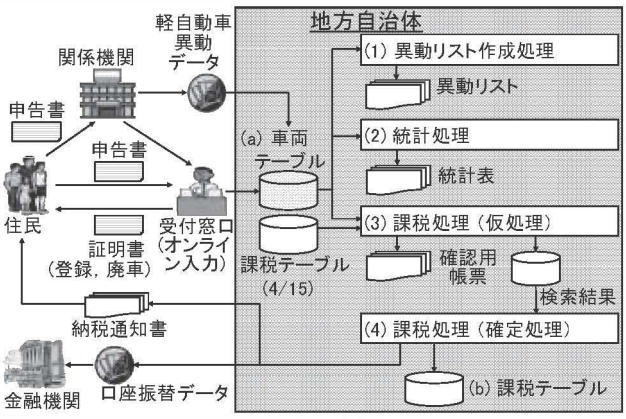

図7　軽自動車税の修更正業務

(a) 車両テーブルのデータ (4/15)

ID	異動日	所有者	異動事由	Ta	Td
001	1/1	軽自太郎	新規登録	1/10	now
002	1/1	廃車次郎	新規登録	1/10	now
003	1/1	譲渡三郎	新規登録	1/10	now

(b) 車両テーブルのデータ (5/16)

ID	異動日	所有者	異動事由	Ta	Td	(A)	(1)	(2)	備考
001	1/1	軽自太郎	新規登録	1/10	now	*	*		
002	1/1	廃車次郎	新規登録	1/10	now	*	*		
002	3/31	廃車次郎	廃車	4/20	5/16				訂正前
002	4/1	廃車次郎	廃車	5/16	now		*	*	訂正後
003	1/1	譲渡三郎	新規登録	1/10	now	*	*		
003	4/1	購入四郎	購入	4/20	5/16				購入→取消
004	4/1	新規五郎	新規登録	4/20	now		*	*	
005	4/1	追加六郎	新規登録	5/16	now				締切り後

図8　異動リスト作成処理における訂正検索

ID	異動日	所有者	異動事由	Ta	Td	(A)	(B)	(1)	(2)
001	1/1	軽自太郎	新規登録	1/10	5/16	*			
002	1/1	廃車次郎	新規登録	1/10	now	*	*	*	
002	3/31	廃車次郎	廃車	4/20	5/16				
002	4/1	廃車次郎	廃車	5/16	now			*	*
003	1/1	譲渡三郎	新規登録	1/10	5/16	*			
003	1/1	購入四郎	新規登録	5/16	now		*	*	
003	4/1	購入四郎	購入	4/20	5/16				
004	4/1	新規五郎	新規登録	4/20	now			*	*
005	4/1	追加六郎	新規登録	5/16	now				

図9　統計処理における訂正検索

更生業務は，その後の免除申請，申告遅れなどに伴う税額の修正や，課税誤りに対する訂正を行うものであり，軽自動車税では月次で行われる．

図 7 に軽自動車税の修更正業務のデータフローを示す．軽自動車の所有に関する異動の申告は，取得が 15 日以内，廃車，譲渡が 30 日以内に行うことが定められており，自治体の窓口で受付けられる他，軽自動車協会，陸運局，商店（以下，関係機関と記載）でも受付けられて自治体に送付される．従って，実世界の異動は車両テーブル(a)に即時には反映されない．このため，修更正は設定された締切り時刻までに入力された申告データを対象に行われた．ここで，自治体の窓口では，申告内容を反映した標識交付証明書や，廃車申告受付書を即時に発行する必要があり，業務時間中はオンライン入力を停止できない．従って，効率的なシステム運用には，オンライン入力中にバッチ処理を実行できることが必要であった．

以下に 4 月 15 日の課税に対する修更正について，図 7 に示した各々の処理におけるバッチ検索処理を示す．なお，以下の説明では，修更正の締切り時刻を 5 月 15 日，バッチ処理の時刻を 5 月 16 日としている．

4.2.1 異動リストの作成処理

図 7 (1)の異動リスト作成処理では，申告データが正しく入力されていることを確認するために異動リストを出力する．軽自動車税のオンライン入力は，申告による異動と，入力などの誤りに対する訂正に分けられた．図 8 の(a)に 4 月 15 日現在，(b)に 5 月 16 日までの申告による異動，および訂正を入力した車両テーブルの状態を示す．車両テーブルでは，式(2)のトランザクション時間以外の主キー属性集合は，$K = \{車両番号, 異動日, 所有者\}$ となる．なお，車両番号は図では「ID」で示し，トランザクション時間T_a，T_dは月日のみを示している．

申告に基づくオンライン入力として，車両番号 002 の廃車（異動日：3 月 31 日），003 の譲渡に伴う購入，004，005 の新規登録が入力されている．一方，訂正はシステム内の操作であるため，削除の場合には終了時刻T_dに削除された時刻が追加され，変更の場合には，さらに訂正後のデータが追加される．図では，5 月 16 日に訂正により$ID = 002$ の異動日の変更，$ID = 003$の譲渡の削除が行われている．

異動リストは 4 月 16 日から 5 月 15 日までの申告による異動を記載した帳票である．従って，検索するデータは訂正を反映した 4 月 16 日から 5 月 15 日までの異動となる．図 8(b)の(A)欄に 4 月 15 日のスナップショット，(1)欄に検索時刻が 5 月 15 日で訂正検索時刻が 5 月 16 日の訂正検索結果の該当データを示す．また，(2)欄に(1)欄の該当データのうち 4 月 16 日以降の追加分のデータを示す．(2)欄では$ID = 002, 003$に対する 5 月 16 日の訂正が反映され，かつ 5 月 16 日に入力された異動$ID = 005$ は反映されない．

すなわち，訂正検索により，異動リストに掲載すべき訂正を反映した 4 月 16 日から 5 月 15 日までに入力された異動データが検索できた．

4.2.2 統計処理

図 7 (2) の統計処理では，5 月 15 日現在の登録車両台数の統計表を出力する．この時，4 月 15 日の統計表との差異が 4.2.1 節の異動リストと整合するか確認することで，データの誤りを検出する．しかし，4 月 15 日以前の入力データに対する訂正が発生した場合には，これらが整合しなくなる．

図 9 に，図 8 に示す状態から，さらに 4 月 1 日に入力済であったデータの訂正として$ID = 001$の削除，$ID = 003$の変更を行った例を示す．(A)，(1)，(2)欄は図 8 と同様の項目であるが(1)欄は上記の訂正の入力後の状態となる．(B)欄 は検索時刻が 4 月 15 日，訂正検索時刻

が5月16日の訂正検索の該当データを示す．4月15日のスナップショットの(A)欄 と5月15日の訂正検索結果(1)欄 の差異は，異動リストの(2)欄と整合しない．しかし，4月15日の訂正検索結果である(B)欄を使用することにより，(B)欄と(1)欄の差異$ID = 002,004$を異動リストの(2)欄に整合させることができた．

4.2.3 課税処理

課税処理では5月15日時点の税額を計算し，4月15日の課税処理の税額から減額されている場合には還付を，増額されている場合には追加徴収を行う．手順としては，図7(3)の課税処理（仮処理）による確認を行った上で，(4)の課税処理(確定処理)で課税テーブル(b)の更新，および住民へ送付する納税通知書と金融機関へ送付する口座振替データを出力する．

図9の例では，修更正の対象は(A)欄の4月15日現在の状態と，(1)欄の5月15日の訂正検索結果の差異になる．すなわち，$ID = 001$は課税取消しによる還付，$ID = 002$は基準日（4月1日）で廃車されていることによる還付，$ID = 003$は所有者訂正により旧所有者への還付，新所有者の追加徴収，$ID = 004$は申告遅れによる追加徴収が行われた．

すなわち，基準日における訂正を反映しないスナップショットと，修更正時点の訂正検索結果の差異を使用することにより，追加徴収，還付の対象を把握することができた．

4.3 訂正検索の実装

3.1節に示したように，訂正検索はトランザクション時間データベースを対象とする．トランザクション時間データベースは，商用のリレーショナルデータベースを使用して，テーブル毎に業務における必要性に応じて追加時刻，削除時刻を付加し構成した．ここで，トランザクション時間の単位はデータベース上の主キー属性となるため，データの更新頻度から設定する必要がある．本システムでは画面からのデータ入力に数秒を要することから，1秒単位とした．なお，トランザクション時間や，これに伴うデータ

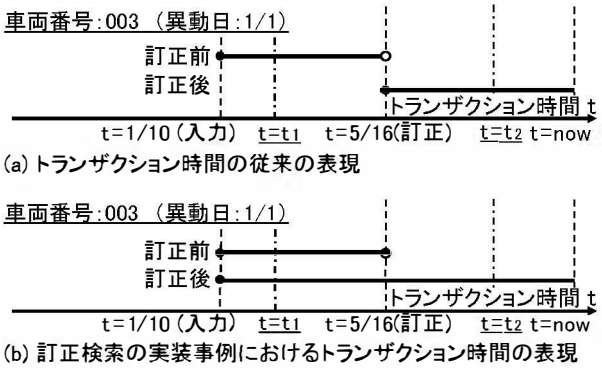

図10 トランザクション時間の実装

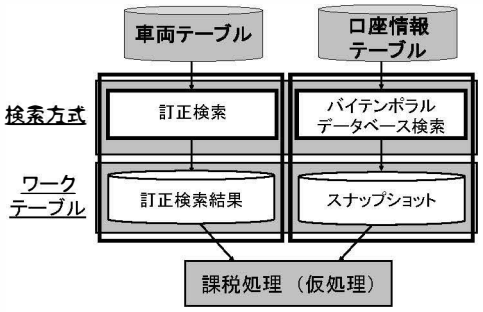

図11 他の検索方式との併用

の履歴はユーザには非公開とし，ユーザからは検索時点の最新の状態のみが検索できる構成にした．

なお，軽自動車税の業務では関係機関からの送付データはオンライン入力の他に，媒体での一括入力も併用した．この場合には，入力データの追加時刻は媒体毎に同一時刻としてデータベースに追加する方式とした．これは，データベースへの個々のデータの挿入に関するコミットを一括して行ったことによる．

トランザクション時間によりデータの変更履歴を表現する場合，従来は図10の(a)に示すように変更後のデータは変更時刻を追加時刻としていた．訂正検索の実装では訂正前後のデータを対応付ける必要があり，従来の表現では検索が複雑化する．この問題に対しては，図10の(b)に示すように，変更前のデータの追加時刻をそのまま変更後のデータの追加時刻とする形式の表現で実装した．

なお，トランザクション時間データベースでは履歴を管理するため，トランザクション時間をテーブルの主キー属性に追加する必要がある．この属性は，いずれの表現においても削除時刻T_dを使用した．これは，新規に追加されるデータは全て$T_d = now$であるため，キー制約を利用して重複データの入力を抑止する機能を容易に実装できることによる．

また，実際の業務システムでは，さまざまな条件でのデータベースの検索が必要になる．例えば，軽自動車税の納税は納税通知書あるいは口座振替によって行われるが，口座振替に関する口座情報，およびその口座の使用期間は住民の依頼に基づく．すなわち，例えば4月からの口座振替開始が車両登録の際などに事前に依頼されるため，繁忙期の入力を避けるためには依頼時点で登録しておく必要がある．これは実世界の有効期間情報であるため，図3に示すバイテンポラルデータベースの構成で実装する必要があった．また，時系列にデータを管理する必要のないマスタデータは，スナップショットデータベースの構成とし，マルチバージョン同時実行制御により検索した．このように，業務によりテーブルの構成が異なるため，さまざまな検索結果を統合し，帳票などの最終的な成果物を作成する必要があった．

このため，適用システムではバッチ処理の中間ファイルはデータベースのワークテーブルで構築し，データベース

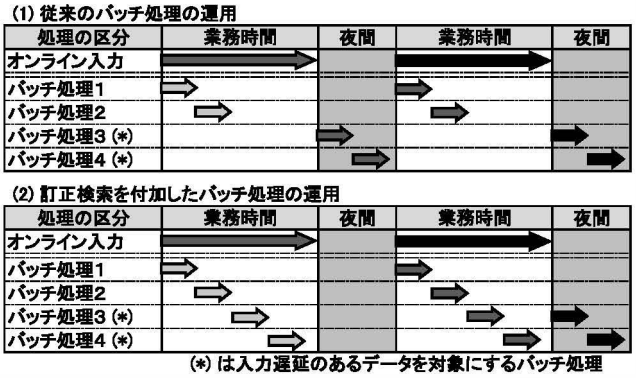

図 12　訂正検索処理による夜間バッチの削減

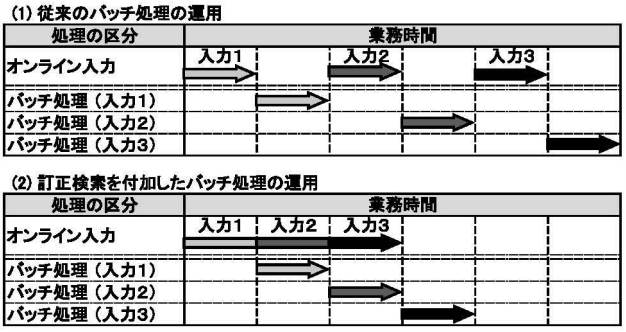

図 13　連続オンライン入力における作業時間短縮

の検索処理だけでなく，それ以降の処理でもデータベースの検索機能を使用して効率的な処理を行う構成にした．例えば，上記の軽自動車税の納税通知書については，図 11 に示すように，車両テーブルは訂正検索で，口座情報はバイテンポラルデータベースのスナップショットで個別に検索し，仮処理でデータベースの検索機能を利用してこれらの検索結果を結合する構成とした．

5. 評価

5.1 バッチ処理の運用の評価

実際のシステム運用においても，訂正検索により 2.2 節に示した課題が解決できた．図 2 のオンライン入力中のデータ訂正に関しては，訂正検索により修更正処理で必要になった以下の訂正と再検索を全て行うことができた．

- **異動リスト作成処理**：訂正を反映した，前回の締切り時刻から今回の締切り時刻までの入力データの再検索．
- **統計処理**：訂正を反映した，前回締切り時刻での再検索と今回締切り時刻での再検索．
- **修更正処理**：前回締切り時刻での再検索と，訂正を反映した今回締切り時刻での再検索．

また，申告の遅れや，関係機関を経由した申告により，入力の遅れるデータが発生したが，訂正検索は締切り時刻までに入力されたデータを検索対象とする方式のため，入力の遅れはバッチ検索処理の問題にはならなかった．

適用システムでは図 7 に示すように，業務時間中は窓口で受付けた情報をオンライン入力し，即時に証明書などが発行される．従って，業務中はオンライン入力を停止できない．一方で訂正を伴うバッチ処理は，入力遅れのデータがある場合には従来はオンライン入力中に実行できないため，オンライン入力の時間帯を避けて実行する必要があった．このようなバッチ処理は，図 12 の(1) に示すように，業務時間終了後に夜間バッチとして実行されていた．適用システムでは図 12 の(2) に示すように，訂正検索により翌日以降の業務時間内に実行する運用が可能になったため，夜間バッチを削減できた．

さらに，業務によっては特定の期間に大量のデータ入力が発生する．例えば，軽自動車税では課税前の 2 月から 3 月に関係機関や住民の申告が行われ，システムへの入力作業のピークになる．業務としては，オンライン入力とバッチ処理による確認が繰り返されるが，従来は図 13 の(1) に示すようにバッチ処理完了後に次の入力を行う必要があった．訂正検索により，図 13 の(2)に示すようにオンライン入力を中断することなく，トランザクション時間により入力された期間を指定してバッチ処理を行うことが可能になった．この結果，オンライン入力とバッチ処理を並行して実行し，作業時間を短縮することができた．

5.2 実装に関する評価

訂正検索は，図 4 に示すように検索処理に実装されるため，テーブルやオンライン入力処理の変更は不要だった．さらに，トランザクション時間を図 10 の形式で実装した結果，訂正検索の機能を容易に実装できた．例えば，図 9 の訂正検索は，式(6)に示す SQL で容易に検索できた．このような実装により訂正検索による性能上の問題は発生しなかった．

$$\text{select} * \text{ from } \textit{車両テーブル} \quad \text{where } r[T_a] < 5/16 \text{ and } r[T_d] = now \tag{6}$$

また，実際の業務運用では，業務によりさまざまな条件でデータベースを検索する必要がある．これに対しては，ワークテーブルによる実装により，図 11 に示すように業務の必要に応じてテーブル毎に適当な検索方法を選択することができた．

5.3 適用に関する評価

適用システムにおいて，訂正検索を適用したテーブル件数，および適用割合を表 1 に示す．トランザクション時間による履歴管理を行う必要があるテーブルには追加時刻T_aを付加し，このうち訂正の発生するものに削除時刻T_dを付加した．従って，表 1 のT_d欄が訂正検索の適用割合になる．ここで，行番号は 4.1 節の業務区分に対応している．なお，以下のテーブルは除いている．

- パラメータ，コードなどを格納したマスタ・テーブル
- 一時的なデータを保存したワークテーブル

表1 訂正検索の適用割合

No	業務区分	テーブル数	Ta	Td
(a)	住民情報系	36	32(89%)	18(50%)
(b)	税関連	72	58(81%)	31(43%)
(c)	福祉系	40	37(93%)	24(60%)
(d)	内部情報系	63	42(67%)	8(13%)
	総計	211	169(80%)	81(38%)

• 集計結果などの導出データのテーブル

追加時刻を持つテーブルは全体の80%であり，訂正検索の適用割合は38%，すなわち履歴管理を行うテーブルの約半数が対象になった．また，適用割合は業務の区分により大きく異なり，内部情報系システム以外では43%から60%のテーブルに適用されたが，(d)の内部情報系システムでは13%に留まった．

4.4節に示したように，実際のシステム運用ではさまざまな条件での検索が発生する．時間軸に関する検索条件に対する検索方式の評価を表2に示す．表2で，「○」は指定された条件でオンライン入力中に一貫性のある検索ができることを示す．また，「×」は一貫性のある検索ができない場合があることを示す．時間を管理しないスナップショットデータベースで指定時刻現在のデータを検索するためには，マルチバージョン同時実行制御が必要になる．ただし，訂正を伴うバッチ処理を行うためには，トランザクション時間データベース，あるいはバイテンポラルデータベースの時間属性を付加し，他の検索方式で検索する必要がある．また，指定された有効時間に関する検索にはバイテンポラルデータベースが必要になるが，図3に示すように，データの入力に遅れがある場合にはスナップショットだけでは，指定されたトランザクション時間における訂正を反映した検索ができないという課題がある．

提案した訂正検索により，この課題が解決できた．ここで，バイテンポラルデータベースにおいてもトランザクション時間を管理しているため，これらのテーブルに対しても訂正検索を適用することができた．また，有効時間はユーザが管理する必要があるため業務によって要否が決定されるのに対し，トランザクション時間はユーザには公開されない．従って，スナップショットデータベースに対しても，システム運用上の必要性に応じて実装できた．

6. 考察

訂正検索によって，指定時刻現在のデータベースを，それ以降の訂正を反映した状態で検索できる．これを実際のシステムに適用した結果，締切り時刻までに入力されたデータを対象とするバッチ検索処理において夜間バッチを削減する効果を確認できた．なお，従来のシステムで行われていた夜間バッチの処理時間は，例えば4万人規模の自治体で1日平均1時間半程度であった．近年は，電子申請，電子商取引などの進展で，利用者が直接オンライン入力し，かつノンストップでサービスを行う運用が拡大している．このため，バッチ処理は従来の夜間バッチからオンライン入力中への実行という運用形態に移行していく必要がある．従って，オンライン入力を停止することなくバッチ処理を実行できる訂正検索は有効であると考えられる．

さらに，ノンストップ・サービスにおけるバッチ処理では，指定された時刻までにオンライン入力されたデータを対象として処理を行う必要がある．この時，誤ったデータに対しては訂正入力を行った上で再処理を行うため，業務による入力データと，誤り訂正による入力データを分けて管理する必要がある．訂正検索では訂正分のみを検索結果に反映するため，締切り時刻までに入力された業務の異動データを対象として，訂正入力とバッチ検索処理による確認を繰り返し行う運用が可能になった．

また，訂正検索はトランザクション時間を管理するテーブルに適用できる．従って，バイテンポラルデータベースによって訂正を反映した検索を行う方式に比較した場合，有効時間を管理する必要がないため，より広い範囲に適用可能であると考えられる．さらに，適用システムの運用では実世界の状態が即時にシステムに入力されないだけでなく，他システムから受領したデータの入力で，バッチ処理による一括入力も併用された．訂正検索は，このようなデータ入力の運用にも適用できることが分かった．

実際の業務システムでは，業務の内容によりさまざまなデータ管理，およびデータに対する検索が必要になる．従って，個々の処理の中でも複数の方式による検索結果を統合し，最終的な出力結果を得られることが重要になる．特に，履歴を含むデータベースに対しては検索処理が複雑化するため，個々の検索を単純化した上で，これらの検索結果を統合した処理を行う必要があった．この点で，トラン

表2 検索条件による適用の評価

データベースの区分	検索方式	オンライン入力中の検索結果		
		指定時刻の検索結果	指定有効時間での訂正を反映した検索	指定トランザクション時間での訂正を反映した検索
スナップショットデータベース	マルチバージョン同時実行制御	○	×	×
バイテンポラルデータベース	スナップショット	○	○	×
トランザクション時間データベース	訂正検索	○	×	○

ザクション時間の追加時刻を最初の追加時刻とした実装や，中間ファイルをワークテーブルとしてデータベースの機能を活用しながら段階的にデータを処理していく方式は有効であると考えられる．

テーブル毎の訂正検索の要否は，表1に示すように業務の区分によって大きく異なった．内部情報系システム以外では，軽自動車税に示すように実世界の申告書に基づくデータを扱うため，申告書の入力による異動と入力誤りなどの訂正を分けて管理する必要があった．一方，内部情報系システムでは庁内業務のため両者を分けておらず，例えば，一旦決裁された伝票に対しては，新たな伝票発行による精算によって訂正する運用が行われた．すなわち，訂正検索は実世界の異動と，内部処理の訂正を分けて管理する必要がある場合に有効であると考えられる．

7. むすび

インターネットの進展によるノンストップ・サービスの広がりにより，指定時刻までに入力されたデータを対象としてバッチ処理を行う運用が広く行われている．しかし，データの訂正が発生する場合には，従来のマルチバージョン同時実行制御や，バイテンポラルデータベースのスナップショットによる検索では，バッチ検索処理で一貫性のある結果を得られない場合があった．

本論文では，指定時刻までに入力されたデータに対して，指定時刻現在の検索結果に，それ以降の訂正を反映する訂正検索を提案した．さらに，これを基幹系システムに適用し，実際のシステム運用でも夜間バッチ処理を削減する効果があることを確認した．

今後の課題としては，オンライン入力とバッチ処理による大量のデータ更新を同時に実行し，データ更新に伴う夜間バッチ処理を削減するための更新方式の実現がある．

参考文献

1） 喜連川優（監訳），J. Gray and A. Reuter（著），*トランザクション処理 －概念と技法―（上）*（日経BP社，東京，2001），pp. 262.

2） 大和田尚孝，渡辺 享靖，小原 忍，“解体! レガシー・バッチ”，日経コンピュータ，No. 659 (2006)，pp.32-47.

3） P. A. Bernstein and N. Goodman, “Multiversion Concurrency Control-Theory and Algorithms”, ACM Trans. on Database Sys. , Vol. 8, No. 4 (1983), pp. 465-483.

4） C. S. Jensen and R. T. Snodgass, “Temporal Data Management”, IEEE Trans. knowledge and Data Eng., Vol. 11, No. 1 (1999) , pp. 36-44.

5） C.S. Jensen, C.E. Dyreson and et al. “The Consensus Glossary of Temporal Database Concept – February 1998 Version”, Temporal Database: Research and Practice, *Lecture Notes in Computer Science 1399* (Springer–Verlag, 1998), pp. 367-405.

6） R. Snodgrass and I. Ahn, “Temporal Databases”, IEEE COMPUTER, Vol. 19, No. 9 (1986), pp. 35-42.

7） N. Edelweiss, P. N. Hübler, M. M.Moro and G.Demartini: “A Temporal Database Management System Implemented on top of a Conventional Database”, Proc. XX International Conference of the Chilean Computer Science Society, (2000), pp. 58–67.

8） G. Özsoyoğlu and R. T. Snodgrass, “Temporal and Real-Time Databases: A survey”, IEEE Trans. knowledge and Data Eng., Vol. 7, No. 4 (1995), pp. 513-532.

9） T. Kudou, M. Ishino, K. Saotome, N. Kataoka and T.Mizuno, “Implementation of Integrity Maintenance Method of Query Result by Bitemporal Database”, International Journal of Infomatics Society, Vol. 1, No.1, (2009), pp. 16-26.

10） A. Motro, “Integrity = validity + completeness”, ACM Trans. on Database Sys., Vol. 14, No. 4 (1989), pp. 480-502.

11） 増永良文，*リレーショナルデータベース入門 －データモデル・SQL・管理システム－*（共立出版，東京，2003），pp. 26，39.

12） L. Shrira and H. Xu: x SNAP, “Efficient Snapshots for Backin-Time Execution”, Proc. 21st International Conference on Data Eng., (2005), pp. 434-445.

13） L. Bækgaard and L. Mark, “Incremental Computation of Time-Varying Query Expressions”, IEEE Trans. Knowledge and Data Eng., Vol. 7, No. 4 (1995-8) , pp. 583-590.

14） B. Stantic, J. Thornton and A. Sattar, “A Novel Approach to Model NOW in Temporal Databases”, Proc. 10th International Symposium on Temporal Representation and Reasoning and Fourth International Conference on Temporal Logic, (2003), pp. 174-180.