時制データベースによるオンライン入力中のバッチ処理方式の研究

静岡大学　博士論文

時制データベースによる
オンライン入力中のバッチ処理方式の研究

平成20年6月

大学院　理工学研究科

設計科学専攻

工 藤　司

論文要旨

基幹系システムではオンライン入力によってデータベースを更新し，バッチ処理で検索して統計資料などを作成するという運用が広く行われている．ここで，オンライン入力では多数の端末から同時にデータを入力するため，トランザクション処理による同時実行制御が行われる．トランザクション処理では，レコード単位などの小さな単位でデータベースの排他制御を行い，個々のトランザクションの応答性能と，データベースの一貫性の双方が維持されるように制御が行われる．一方，バッチ処理では個々の処理の応答性能よりも全体のスループットが重視され，大量のデータを効率良く処理するために，データベースはテーブル単位などの大きな単位でジョブに割り当てられる．このように，両者は異なった処理形態で構成されているため，従来はバッチ処理を実行する時間帯はオンライン入力の時間帯と分けられ，オンライン入力を伴う業務終了後に夜間バッチとして実行されていた．しかし，近年では基幹系システムの業務範囲の拡大や，インターネットの進展に伴う電子商取引などの Web システムの普及によりオンライン入力を伴うサービス時間の延長が必要になっており，夜間バッチが所定の時間内に完了しなくなるという問題が発生している．

これに対し，マルチバージョン同時実行制御を始めとして，両者を並行して実行するための方式が提案されている．しかし，従来の方式では実際のシステム運用で発生する，データ訂正を伴うバッチ処理の再実行や，特定期日への処理集中などの問題に対応できないという課題があった．一方，時制データベースは時系列に変化するデータを管理するデータベースであり，このうち，ある事実がデータベース内に存在していたトランザクション時間と，ある事実が実世界において有効だった有効時間の，双方に関する時系列のデータ履歴を管理するデータベースはバイテンポラルデータベースと呼ばれる．バイテンポラルデータベースでは，オンライン入力中にもデータ訂正の結果を反映した一貫性のある検索結果が得られる．しかし，バイテンポラルデータベースでは 2 種類の時間属性を管理する必要があり，検索機能が複雑化すると共に履歴によるデータ量の増大が発生するため実装の事例が少なく，特に基幹系システムに適用，評価した事例はない．

本研究では，バイテンポラルデータベースにより，基幹系システムにおいてオンライン入力と並行してバッチ処理を行うための方式を提案する．まず，バイテンポラルデータベースにより，オンライン入力中であっても，実際のシステム運用において発生する訂正が反映された一貫性のある検索が可能であることを示す．さらに，これを基幹系システム

に適用した結果を評価し，従来，特定期日に夜間バッチで実行されていた処理を，翌日以降のオンライン入力中に実行することでシステム運用負荷を削減する効果があることを確認した．一方，適用の結果として，遅れて入力されたデータが検索から漏れる場合があることが分かった．

そこで本研究では，この課題に対する改良として訂正検索を提案する．訂正検索では，定期的なバッチ処理が締め切り時刻までに入力されたデータを対象にすることを利用し，締め切り時刻現在の検索結果に，それ以降に行った訂正を反映したものを検索結果とする．訂正検索により，バッチ処理において遅れて入力されたデータの検索漏れを防止できること，バイテンポラルデータベースの従来の検索機能と訂正検索の双方を併用したバッチ処理が構成できることを示す．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発生する検索においてもバイテンポラルデータベースの課題が解決できることを確認した．

バイテンポラルデータベースを基幹系システムに適用・評価した事例は，本論文が初である．適用にあたっては，バッチ処理の中間ファイルをデータベースの一時的なテーブルで構成して段階的にデータを処理することで検索を単純化し，実世界の履歴をイベント型の履歴によって表現することで履歴データの増加を抑制した．この方式により商用のリレーショナルデータベース上に構築したバイテンポラルデータベースを基幹系システムに適用して，実際のシステム運用において夜間バッチ削減の効果を確認した．また，適用によって得た知見として，長期間に渡るデータ訂正を行う業務でもデータの履歴管理が容易になること，内部処理での訂正と業務処理での訂正を分けて管理できること，複数の時刻現在のデータを統合した処理が実行できることを確認した．さらに，訂正検索を付加することにより，オンライン入力だけでなく一括入力されたデータにも適用可能になることを確認した．

今後，Web システムが進展していくにつれ，基幹系システムでもノンストップのオンライン入力サービスが広がっていくと考えられる．このようなシステムでは，オンライン入力とバッチ処理の時間帯を分けた運用を行うことができない．従って，本研究の提案した，バイテンポラルデータベースおよび訂正検索による基幹系システムの構築は有効であると考えられる．

目 次

図 目 次

表 目 次

第1章　序論

1.1　研究の背景及び目的

情報システムは製造，販売，行政，医療などのさまざまな業務分野で広く活用されており [54, 57, 62, 79, 95]，多くの業務が情報システムに深く依存するようになっている．この結果，業務の遂行を支援するための情報システム（以下，業務システムと略記）は，企業などの競争力や，業務の遂行そのものに大きな影響を与えるようになっている．

業務システムは，企業などにおける本来の業務に直結している基幹系システムと，これを支援する情報系システムに分類される [53]．図 1.1 に小売業における POS（Point Of Sales）システム [53] を活用した業務システムの事例を示す．POS システムでは，店舗の商品販売情報を記録し，販売情報データベースに蓄積することにより，いわゆる販売時点管理が行われる．販売情報データベースを関連するさまざまな業務で活用することにより，広範囲の業務を一元的に遂行することができる．図 1.1 の事例では，在庫・受発注管理，会計管理や決済といった業務システムで販売情報データベースに基づき処理が実行される．これらは，企業本来の業務に直結しており，基幹系システムに分類される．一方で，販売情報データベースは複数店舗の比較や時系列による販売動向の把握など，マーケティングや意思決定のためのデータ分析処理 [12, 58, 81, 88, 52]，業績管理 [84] などの販売情報を活用した企業内 OA（Office Automation）処理，あるいは外部への情報公開などで活用される．これらの業務システムは，企業本来の業務を支援する情報系システムに分類される．これらのシステムの処理を比較した場合，基幹系システムでは日常の業務を遂行するため比較的定型的な処理が多く，情報系システムではデータ分析などの非定型的な

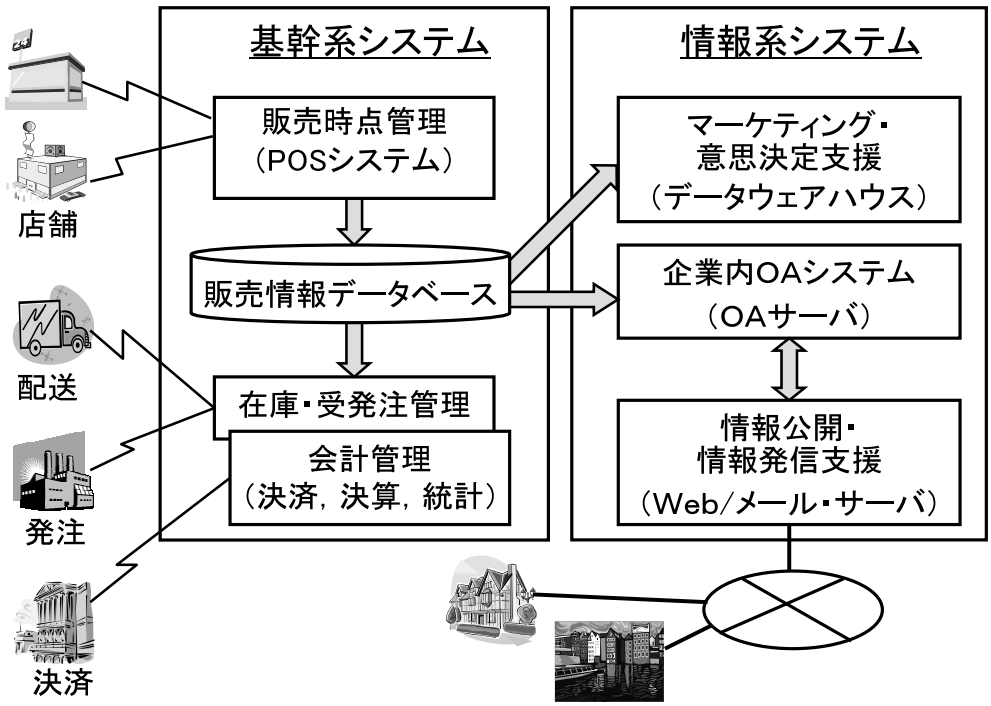


図 1.1: ＰＯＳシステムを利用した業務システムの構成事例

処理が多いという特徴がある．

ここで，基幹系システムでは，データをオンライン処理により業務端末から入力（以下，オンライン入力と略記）してデータベースに蓄積し，定期的に決算資料や発注データなどを作成するという運用が広く行われている．例えば，POS システムでは業務時間帯に売上情報が各店舗でオンライン入力されてレシートが発行され，入力データはデータベースに蓄積される．そして，当日や月末などの業務終了時点で，受注の配送指示や，発注処理，あるいは会計業務における決算処理などが行われる．

オンライン入力は同時に多数の端末から対話型で実行される．従って，各々のユーザが他の端末のユーザを意識することなく操作でき，また，個々の操作が充分な応答性能を持つ必要がある．一方，統計資料などの作成を行う場合には，大量のデータを処理するために時間を要し，非対話型のバッチ処理で実行される [85]．バッチ処理ではさまざまな処理を一括して実行するため，個々の処理の応答性能よりも，全ての処理が完了するまでに要する時間が重要になる．従って，大量のデータを効率的に処理するために，データベースはテーブル単位などの大きな粒度で処理に割り当てられ，同時に実行する処理の数は制限されて全体としてのスループットの向上が図られる．このようにオンライン入力とバッチ処理では要件に相違があり，異なった方式で実行される．表 1.1 にオンライン入力とバッ

表 1.1: オンライン入力とバッチ処理の特徴比較

No	項目	オンライン入力	バッチ処理
(1)	処理内容	端末からのデータ入力・照会	入力済データの一括処理
(2)	処理形態	対話型	非対話型
(3)	データ量	少量	大量
(4)	処理時間	１件毎に短時間で完了	全体として所定の処理時間で完了
(5)	占有リソース	小さい（レコード単位など）	大きい（テーブル単位など）
(6)	同時実行	多い	少ない
(7)	処理単位	トランザクション	ジョブ

チ処理の特徴の比較を示す.

オンライン入力では多数の入力が同時に行われるため，同時実行制御 [70, 90, 101] によりデータベースの一貫性を維持することが重要になる．ここで,「一貫性」は，データベースが対象となった実世界の事物の状態を正しく表現していることと定義する [100]．同時実行制御は，通常データベース管理システム（DabaBase Management System，以下，DBMS と略記）に実装され，DBMS 上での処理単位はトランザクション（transaction）[70, 73, 89, 97] と呼ばれる．トランザクションは同時実行制御により以下の ACID 特性を維持するように制御される [94].

- **Atomicity（原子性）** トランザクションの終了時，全ての処理が完了しているか，全く実行されないかのどちらかである.
- **Consistency（一貫性）** トランザクションの終了状態にかかわらず，データベースは一貫性を維持している.
- **Isolation（隔離性）** トランザクションの中間結果を他のトランザクションは見られない.
- **Durability（耐久性）** トランザクションが完了すると，その後の障害などでデータベースの内容が変化しない.

トランザクションは，同時実行制御を行うために操作するリソースを他のトランザク

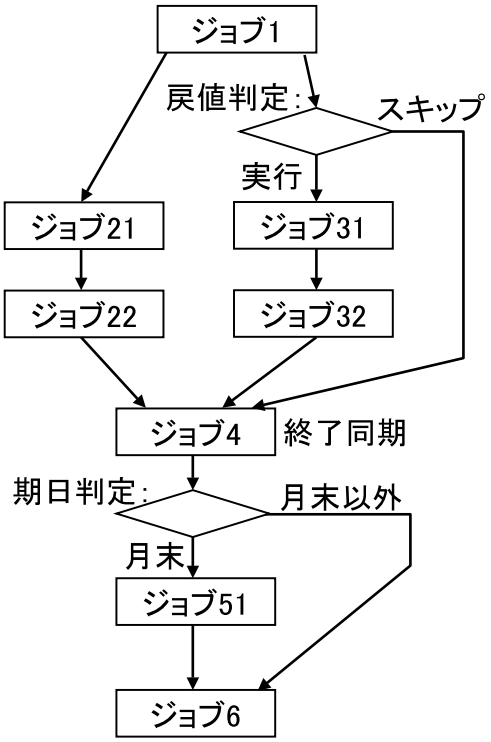
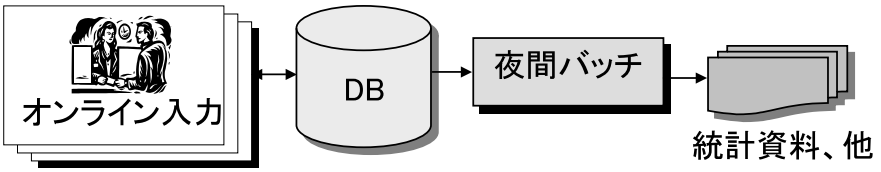


図 1.2: ジョブネットによるジョブの実行制御

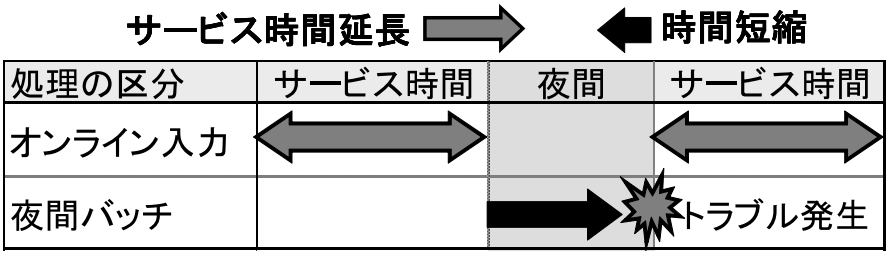
ションに操作させないようにロックして占有するが，オンライン入力では単位時間に処理されるトランザクション数を最大化し，各々のトランザクションの応答性能を維持することもまた重要になる [82]．このため，トランザクション処理では，ロックの単位やロック時間を最小化し，複数のトランザクションが並行実行されると共に，一貫性が維持されるようにトランザクションの実行順序制御が行われる．

一方，バッチ処理の処理単位はジョブ（job）と呼ばれる．バッチ処理では同じ時間帯に実行される処理全体での実行時間を短縮することが優先される．従って，効率的にジョブを実行するために，データベースはテーブルや，ファイルなどの大きな粒度でジョブに割り当てられ，各ジョブはこれを占有して処理を実行する [70]．また，バッチ処理では処理対象のデータ量が多く，効率的にジョブを実行するためには，ＣＰＵ，メモリ，ディスクなどのシステムリソースを充分に割り当てる必要がある．このため，全体の実行時間が最短になるように，同時に実行されるジョブの数を制御する必要がある．さらに，バッチ処理は長時間におよぶため，処理の自動制御・監視もまた必要になる．

このようなジョブの実行制御は，ジョブ管理システムによって行われ，ジョブネットとして表現される [85]．図 1.2 にジョブネットの構成を示す．ジョブネットでは，矢印で結



(a) オンライン入力と夜間バッチのデータフロー



(b) 夜間バッチの処理時間

図 1.3: 夜間バッチの処理時間短縮

ばれたジョブが逐次実行されることで，実行ジョブの順序制御が行われる．また，図 1.2 の例では，ジョブ 31 はジョブ 1 の実行結果により実行有無が制御され，ジョブ 21，ジョブ 22 と，ジョブ 31，ジョブ 32 は並行して実行されて，ジョブ 4 は同期制御により双方の完了を待って開始される．また，ジョブ 51 は期日判定により制御され，月末のみ実行される．このように，同一のリソースを使用するジョブを逐次実行することで競合を発生させずにジョブを並行実行でき，同時実行するジョブの数を制御することで全体としてのスループットを向上することができる．また，各ジョブの戻り値や，月末などの指定期日で実行するジョブを変更することにより，自動的にバッチ処理を実行することが可能である．

バッチ処理では長時間に渡りリソースを占有するため，オンライン入力と同じデータをアクセスする場合には，オンライン入力と時間帯を分けて実行されることが多い．例えば，小売システムでは図 1.3(a) に示すようにデータがオンライン入力され，業務時間終了後の夜間にバッチ処理により入力データを検索して統計資料の作成や，決算処理を行うという運用が行われてきた．しかし，インターネットの進展に伴うユビキタス・コンピューティングの流れの中で，基幹系システムもまた，業務範囲の拡大やサービス時間の延長が必要になっている．例えば，電子商取引や [95, 103]，電子申請 [80, 87, 107] などの Web システムを活用したサービスや，金融業界の ATM による夜間サービスなどが普及し，これに伴い，オンライン入力を必要とする業務時間帯が延長されている．

このため，時間帯を分けた運用ではバッチ処理時間の制約が問題になっている．すなわち，オンライン入力終了後に，バッチ処理は夜間の処理（以下，夜間バッチと略記）として実行され，翌日のオンライン入力開始前に終了する必要がある．しかし，夜間バッチに割り当てられる時間が制約されるため，バッチ処理終了からオンライン入力開始までの時間の余裕が少なくなる．一方，定常的なバッチ処理であっても発生するデータ量は一定ではないため，処理時間は常に変動する．例えば，小売の売上件数は日によって変動するため，件数が多い日には日次決算の処理時間が長くなる．この結果，規定の時間内にバッチ処理が完了しない，いわゆる「つきぬけ」が発生してシステム運用に余裕がなくなり，トラブルが発生しやすくなるという課題がある [61]．このように，オンライン入力時間時間の延長により，オンライン入力とバッチ処理を分離する運用では，図 1.3(b) に示すようにバッチ処理が所定の時間内に完了しなくなるという課題が発生している．

本研究では，基幹系システムのオンライン入力時間の延長に伴う，バッチ処理時間の短縮という課題を解決するため，オンライン入力とバッチ処理でのデータベース検索処理（以下，バッチ検索処理と略記）を並行して実行するための検索方式を示すことを目的とする．この検索方式は，実際の基幹系システムに適用することを目的とし，システム運用で発生するさまざまな問題，例えば，誤った入力データに対する訂正や，月末などのバッチ処理の集中する特定期日の負荷への対応ができることを狙う．

1.2 従来研究の概観

バッチ処理が所定の時間内に完了しなくなるという課題を解決するために，さまざまな方面からの研究が行われている．

1.2.1 バッチ処理実行時間の短縮

ジョブの実行時間短縮の点からは，長須賀らによりジョブ間のデータの受渡しに関する効率化が提案されている [91]．ジョブ間のデータ受渡しにはディスク上のファイルが利用され，先行するジョブが完了した後，後続のジョブが開始される．長須賀らの提案では，ファイルをメモリ上の仮想記憶上に保存することで高速化し，さらにデータの一部が書き込まれた時点で後続のジョブを開始されることで，ジョブの並列性の向上を図っている．また，ジョブネットのスケジューリングに関しては，上西らにより最適化に関する提案が

行われている [55]．基幹系システムのバッチ処理は定型的であり，周期的に実行されるものが多い．上西らの提案は，このことを利用してジョブの実行時間の実績から，統計的に処理時間を予測し，スケジューリングを最適化するものである．しかし，これらの方法では逐次処理を行わなければならないジョブが残されるため，ジョブ全体の処理時間は，逐次実行されるジョブの合計処理時間までしか短縮できない．従って，許容される時間がこの時間より短い場合には，バッチ処理を完了できないという課題がある．

さらに，ジョブの実行自体を高速化する方式として DBMS の高速化 [37]，複数サーバで分散実行して処理時間を短縮する方法 [46, 61]，メモリ共有型のマルチプロセッサを使用する方式 [74]，あるいはメモリ常駐型のリレーショナル DBMS[93] などが提案されている．これらの方法によりある程度のバッチ処理時間の短縮が可能になっている．しかし，許容される時間がこれらの方式による処理時間よりも短い場合には，バッチ処理を完了できないという課題がある．

1.2.2 オンライン入力中のバッチ検索処理の実行

オンライン入力とバッチ検索処理を並行して実行し，バッチ処理時間の制約を回避するの方式が提案されている．例えば，DBMS ではレコード・ロックにより複数のアクセスの一貫性を維持する機能が備えられ，さらに，ミニバッチ（mini-batch）[70] はバッチ検索処理を短時間のトランザクションに分割して実行する方式であり大量のリソースを長時間に渡り占有することを避けることができる．例えば，オンラインでの検索と同様に，1 件毎の検索を行うトランザクションの集合としてバッチ検索処理を構成すれば，オンライン入力と並行して実行することができる．しかし，この方法では検索の途中でオンライン入力による更新が行われるため，検索結果の一部は更新前のデータ，他は更新後のデータとなり，検索結果の一貫性が維持できなくなるという課題がある．例えば，指定時刻時点の集計を行うために，指定時刻から検索を開始したとしても，検索処理が完了するまでには時間を要するため，一部の検索結果は指定時刻以降のオインライン入力によって更新されたデータになってしまう．

また，バッチ検索処理の間に実行されたオンライン入力の影響を受けない検索方式として，マルチバージョン同時実行制御（multiversion concurrency control）[1, 17, 31, 42, 46, 65, 74] の研究が進められ，多くのデータベースで採用されている [94]．これは，トランザクションがデータベースを更新するとき，DBMS がロールバックに備えて採取する更新前のデータを利用して検索を行うものであり，オンライン入力中であっても検索開始時点のデータ

を検索することができる．これと同様の技術を，アプリケーションプログラムで実現した事例も報告されている [59, 60].

しかし，実際の基幹系システムに適用するためには，誤入力されたデータの訂正や，データを訂正した後のバッチ検索処理の再実行，あるいは特定時刻現在におけるデータの検索など，さまざまな運用に対応できる必要がある．例えば，基幹システムは業務と密着しているため，その結果には高い正確さが求められる．オンライン入力に際し，データベースシステムではトランザクションに一貫性制約が課され，これらにより一定の一貫性が確保されるが，多数の端末からの入力に対するレスポンスを確保するため，大量のデータ検索を伴う整合性の確認，例えば複数のテーブルの突合せ，あるいは現金，棚卸し結果と突合せるための集計はバッチ処理によって行われる．従って，バッチ処理は誤ったデータを訂正した上で，処理を再実行できる必要がある．ところが，マルチバージョン同時実行制御では検索開始時点のデータベースの状態を検索するため，データ訂正後にバッチ検索処理を再実行すると，この間に新たに入力されたデータも検索対象になるという課題がある．さらに，検索時刻は検索処理開始時刻として暗に指定されているだけであるため，高い頻度でデータが入力されている場合には，どの時刻までに入力されたデータが検索対象になるかを厳密に指定できない.

特定時刻の状態を検索するための方法としては，データベースの過去の一時点にシステムを復元するための提案がある [31]．過去の一時点におけるデータベースの状態はスナップショット（snapshot）と呼ばれる．提案の方式では，スナップショットはデータベース操作のログにより復元される．さらに，特定のログを抽出して不要な操作を除いて復元する方式 [8] や，ログを効率的に保持するために必要なスナップショットの前後に限定して保持する方式 [86] が提案されている．しかし，これらはデータベースを指定時刻に復元するための技術であり，データ訂正が発生した場合には，オンライン入力された訂正データを反映してバッチ検索処理を再実行するものではない.

データ訂正に関しては，版管理データベース（multiversion database）[44, 50, 69, 77, 83] を使用することで，指定時刻に対する訂正後の版をオンライン入力と別に管理できる．版管理データベースは，時間の経過と共に作成されるデータについて，作成や削除の時刻，版の導出関係などを管理するものであり [29, 69, 77]，時系列のデータ変化だけでなく枝分かれのある版の導出関係が管理できる [69]. 従って，ある時刻のデータ集合を１つの版とするとき，この版に対するオンライン入力によって時系列に作成される版の集合と，この版に対するデータ訂正を行った版の両方が構成できる．また，検索に関しては，検索結

果の一貫性を満たせること [44, 77] や，検索言語 [50]，質問のクラス関する提案 [83] が示されている．版管理データベースはCADやプログラム開発などの，複数のバージョンを持つ設計情報を管理する場合には不可欠なものなっている [69]．しかし，基幹系システムのように，さまざまなタイミングでデータの検索を行うシステムに適用する場合，頻繁に訂正データを検出し新たな版を構成しなければならず，実際的な解決にならないという課題がある．

このように，従来技術によりオンライン入力中にバッチ処理を実行する場合には課題があり，実際のバッチ処理ではこれらの課題を回避するようにシステムを運用する必要がある．例えば，月次のバッチ処理で突き抜けを防止するために，実行期日前に仮に当日の処理を実行しておき，その後の変更データはオンラインで修正する運用方式が報告されている．しかし，同時にこのような運用では重複作業の発生などにより全体としての作業効率が落ちるという課題があることが報告されている [61]．

1.2.3 時制データベースによるバッチ検索処理

近年，時制データベース（temporal database）[28, 33, 35, 38, 39, 49] の研究が進展している．時制データベースは，時系列に変化するデータの履歴を管理するデータベースであり，その時間属性は，ある事実がデータベース内に存在していた時間であるトランザクション時間（transaction time）[18, 28, 33]，ある事実が実世界において有効だった有効時間（valid time）[28, 33]，システム利用者によって定義されるユーザ定義時間（user-defined time）[28, 33] に区分される．トランザクション時間を管理するデータベースはトランザクション時間データベース（transaction time database），有効時間を管理するデータベースは有効時間データベース（valid time database），トランザクション時間と有効時間の双方を管理するデータベースはバイテンポラルデータベース（bitemporal database）と呼ばれる．

このうち，トランザクション時間データベースでは，過去の任意のトランザクション時刻におけるスナップショットを検索することができるため，オンライン中のバッチ検索処理の実行と，指定した時刻時点の状態の検索が可能になる．まず，スナップショットはデータベースにおける過去の時点の状態であり，一方，オンライン入力は現在時点のデータベースの状態を更新する．従って，検索中にオンライン入力が実行されても，その影響を受けずにバッチ検索処理を行うことが可能になる．また，データベースにおける時系列のデータの更新履歴が管理されているため，過去の任意のトランザクション時間を指定し

た検索を行うことができる．例えば，月末時点などの特定期日のデータを，それ以降の任意の期日に検索することが可能になる．

しかし，1.2.2 節に示したように，実際のバッチ検索処理の運用ではデータ誤りが発生する．ところが，トランザクション時間データベースでは指定時刻現在のスナップショットを検索するため，このデータの訂正を行った後でバッチ検索処理の再実行を行っても，指定時刻以降に行われたデータ訂正が反映されない．すなわち，データ訂正を行った場合には検索処理の再実行を行うことができないという課題がある．

一方，バイテンポラルデータベースでは有効時間を指定することで検索対象データを特定し，トランザクション時間にデータの訂正後の時刻を指定することで，データ訂正の結果を反映した状態を検索することができる．ところが，バイテンポラルデータベースでは２種類の時間属性を管理するため，データベース構築のためのデータモデル [101] や，データベースの検索が複雑になる．さらに，２つの時間属性に関する履歴を残す必要があり，データ量が増大するという課題がある．このため，バイテンポラルデータベースでは実装に関する提案は行われているが [35, 11]，実装事例は少なく実際の基幹系システムに適用，評価した事例は見当たらない．

1.3　本研究の位置付けと特徴

本研究は，トランザクション処理によるオンライン入力と，バッチ処理による大量データの検索処理から構成される基幹系システムにおいて，両者の処理時間を分けるという運用上の制約を回避できるデータベースを提供することを目的とする．また，データベースの研究の点からは，バイテンポラルデータベースの実装研究に位置付けられる．具体的には，オンライン入力と並行してバッチ検索処理を実行するための検索方式を，バイテンポラルデータベースによって構築し，これを実際の基幹系システムに適用，評価したこと，および，その結果に基づくバイテンポラルデータベースの検索方式の改良を提案したことを特徴とする．特に，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，評価したのは，本研究が最初の事例である．本研究の特徴と構成，および効果を図 1.4 に示す．

本研究では，まず，バイテンポラルデータベースによりオンライン入力中であっても，実際のシステム運用において発生する訂正が反映された，一貫性のある検索が可能であることを示す．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においても，バイテンポラルデータベースによりオンライン入力とバッチ検索処理を並行して実行で

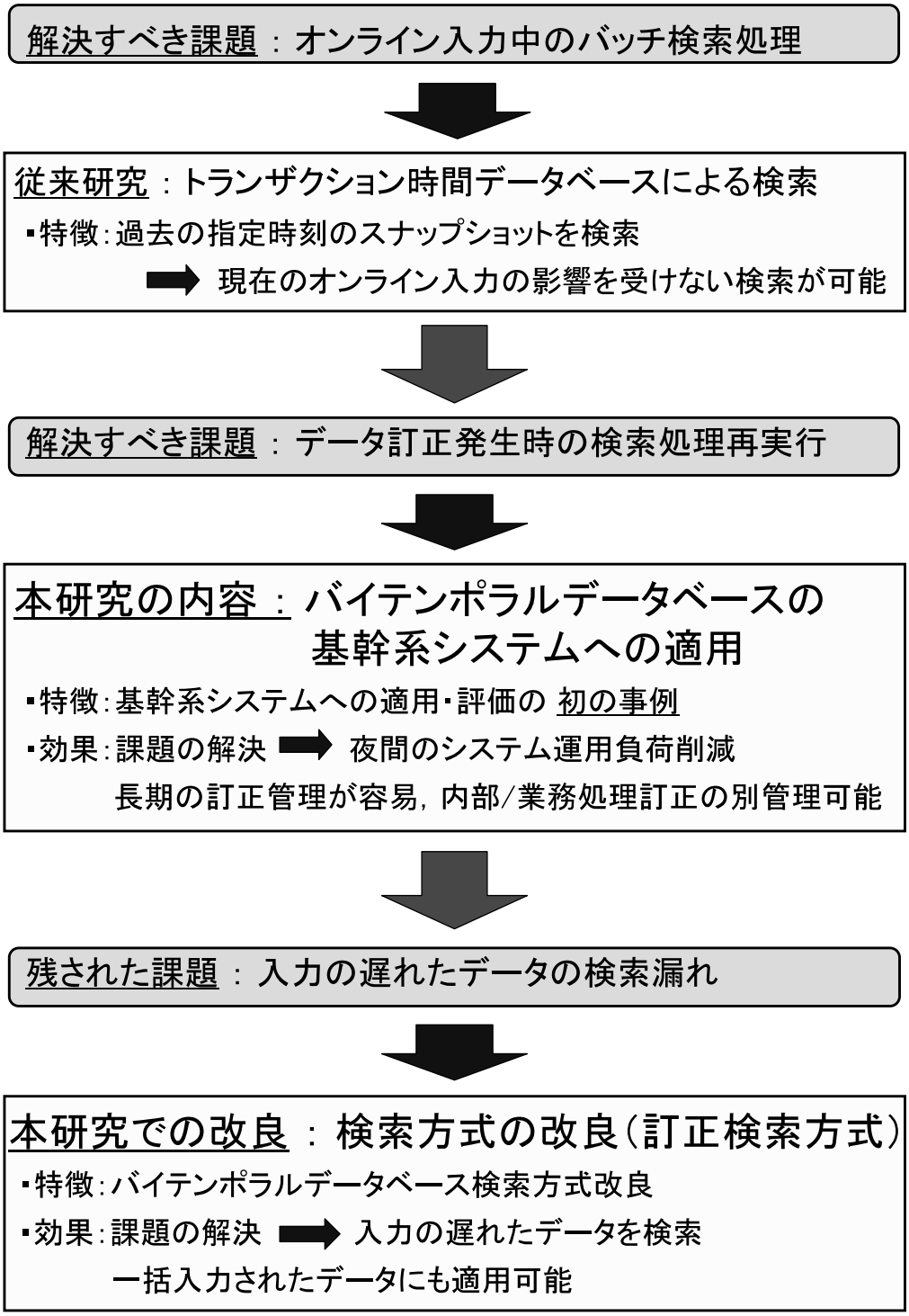


図 1.4: 本研究の特徴と効果

きること，夜間バッチを削減する効果があることを確認した．一方，適用の結果，実際のシステム運用では実世界の状態が入力されるまでに時間を要する場合があり，入力の遅れたデータが検索対象から漏れるという課題があることが分かった．適用システムではユーザ定義時間である届出がなされた日を，有効時間の代替として使用する改良を行った．しかし，これに該当する属性のない場合や，届出がなされた情報が即日入力されない業務では，バッチ検索処理でデータの検索漏れが発生する場合があるという課題が残った．

本研究では，この課題に対し改良した検索方式として訂正検索を提案する．訂正検索では，定期的なバッチ処理が締め切り時刻までに入力されたデータを対象とすることを利用し，締め切り時刻現在の検索結果に，それ以降に発生した訂正を反映したものを検索結果とする．これにより，バイテンポラルデータベースの課題が解決できること，バイテンポラルデータベースの従来の検索機能と訂正検索の両方を利用したバッチ検索処理が構成できることを示す．さらに，これを実際の基幹系システムに適用し，実際のシステム運用でも有効であることを確認した．

また，バイテンポラルデータベースを基幹系システムに適用した事例は本論文が初めてであり，適用によって得た知見について評価，考察する．まず，実装にあたっては 2 種類の時系列データの履歴管理に伴う検索の複雑化や，データ量の増大という課題があった．これに対し，バッチ処理の中間ファイルをデータベースの一時的なテーブルで構成して段階的にデータを処理することで検索を単純化し，実世界の履歴をイベント型の履歴によって表現することで履歴データの増加を抑制した．この方式により，商用のリレーショナルデータベースの上に構築したバイテンポラルデータベースを基幹系システムに適用し，実際に運用できることを確認した．

また，バイテンポラルデータベースを適用した効果として，実際のシステム運用において夜間バッチを削減できるという効果の他に，長期間に渡るデータ訂正を行う業務でもデータの履歴管理が容易になること，内部処理での訂正と業務処理での訂正を分けて管理できること，複数の時刻現在のデータを統合した処理が実行できることを確認した．さらに，訂正検索は，オンライン入力だけでなく一括入力されたデータにも適用可能になることを確認した．

1.4　本論文の構成

本論文は５章で構成される．第１章は序論であり，本研究の背景と目的，および特徴を述べた．第２章では，まず，対象とする基幹系システムのデータフローと，実際のシステム運用においてオンライン入力中にバッチ検索処理を実行するための要件を示す．次に，本論文の基礎となる時制データベースの従来研究を概観し，バッチ検索処理に適用した場合の課題を示す．第３章では，バイテンポラルデータベースによるバッチ検索処理によりこの課題が解決できることを示し，これを実際の基幹系システムである自治体システムに適用した事例について評価，考察する．さらに，バイテンポラルデータベースを基幹系システムに適用することによって得た知見について示す．第４章では，バイテンポラルデータベースにおいて，データのオンライン入力が遅れた場合に発生する課題を示し，課題を解決するための検索方式として訂正検索方式を提案する．さらに，これを自治体システムに適用した結果を示し，これを評価，考察する．第５章では，まとめと，今後の課題について述べる．

第2章　バッチ検索処理における時制データベースの課題

2.1　まえがき

実際の基幹系システムにおいて，オンライン入力中にバッチ検索処理を実行するためには，そこで発生する問題，例えば，誤ったデータを訂正した上でのバッチ検索処理の再実行や，特定の期日のデータを使用する処理が集中する場合の負荷の分散など，実際のシステム運用で発生する問題に対応できる必要がある．そこで，本章では基幹系システムのデータフローと運用について示し，この結果からバッチ検索処理をオンライン入力と並行実行するための方式の要件を示す．

次に，本論文の基礎となる時制データベース（temporal database）の従来研究について概観する．データベースには対象とする実世界の事物の値がデータとして保存されている．ここで，実世界の状態は刻々と変化しているため，時間をサポートしないデータベースでデータを新鮮に保つためには，実世界の変化にあわせてデータベースのデータも常に更新し続けなければならない．このようなデータベースは刻々と変化していく実世界のある一瞬をモデル化したものであると考えられる．この断面はスナップショット（snapshot）と呼ばれ，時間をサポートしないデータベースはスナップショットデータベース（snapshot database）[5, 33, 49] と呼ばれる．これに対し，時制データベースは，時間的に変化するデータの履歴を管理する．このため，データの変更や削除は論理的な操作として行われ，物理的には一旦追加されたデータは削除されずに残される．本章では，まず，時制データベースの研究動向全般について概観し，次いで，本研究に関連する検索方式と実装につい

て概観する．

次いで，オンライン入力中にバッチ検索処理を行う基幹系システムを，時制データベースにより構築する場合について，上記の要件の点から評価する．時制データベースは第１章に示したように，トランザクションデータベース，有効時間データベース，バイテンポラルデータベースに区分される．ここでは，従来から提案されているトランザクション時間データベースによるバッチ検索処理の検索方式と，その課題を示す．この課題は，本研究の対象であるバイテンポラルデータベースや，訂正検索で解決しようとするものである．さらに，有効時間データベースとバイテンポラルデータベースの課題を示す．

本章は，以下のように構成される．2.2 節でオンライン入力と並行してバッチ検索処理を実行するための要件を示し，2.3 節で時制データベースの従来研究を概観する．2.4 節で時間データベースによるバッチ検索処理方式と，各時制データベースの要件に対する課題を示す．最後に 2.5 節でまとめを述べる．

2.2　バッチ検索処理の要件

本論文が対象としている基幹系システムのデータフローを図 2.1 に示す．業務端末からさまざまなデータがオンライン入力され，トランザクションによって即時にデータベースに反映される．この時，業務端末側で出力が必要な帳票があれば，併せてトランザクションによって出力処理が実行される．入力されたデータはデータベースに蓄積されて，定期あるいは随時のさまざまなバッチ処理によって利用される．バッチ処理は，図 2.1 のバッチ検索処理，仮処理，確定処理に区分される．まず，バッチ検索処理ではデータベースから必要なデータを抽出し，検索結果ファイルに出力する．次に，仮処理で検索結果ファイルのデータに対するさまざまなチェックが行われ，チェックリストなどの確認用帳票が出力される．確認作業終了後に，確定処理によってデータの確定や，業務用帳票の出力が行われる．

このような，業務処理におけるデータの一貫性の確認は，オンライン入力と仮処理の両方で行われる．オンライン入力のトランザクションには以下のような一貫性制約（integrity constraint）[100] が課される．

- **ドメイン制約（domain constraint）**
  データが取りえる値の範囲に関する制約．例えば，商品の数量あるいは単価は数値であり，マイナスにならない．

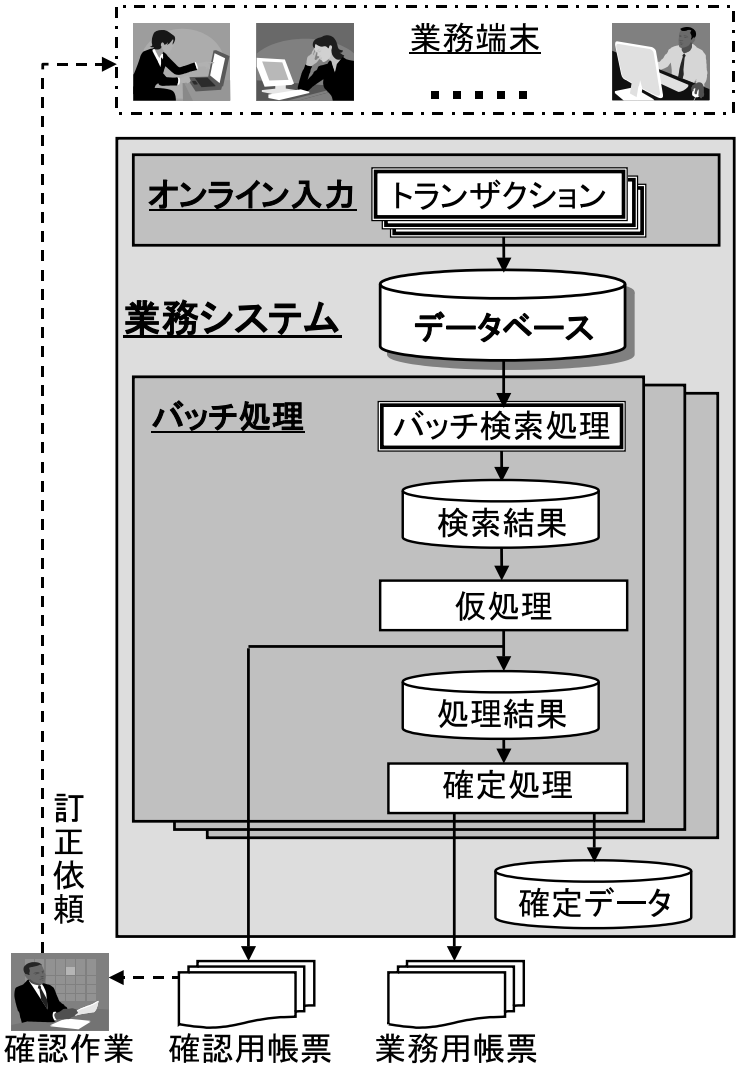


図 2.1: 基幹系システムのデータフロー

- **キー制約（key constraint）**
  同一の主キーを持つデータが重複していないこと．例えば，扱う商品の情報を，以下のリレーション [100] を持つ商品テーブルで管理する場合，同一の商品コードを持つ複数の商品の情報は登録できない．

  商品テーブル（商品コード，商品名，単価）

- **外部キー制約（foreign key constraint）**
  他のテーブルを参照する属性は，他のテーブルに存在するキーの値を取ること．例えば，あるテーブルで商品コードを属性として持つ場合，商品テーブルにない商品

コードを属性値とすることはできない.

さらに，オンライン入力のアプリケーション・プログラムにおいても，該当の入力作業に関する個別の入力チェックが行われ，誤りが検出された場合には入力の訂正要求が画面に表示され，誤入力の防止が図られる．また，複数部門でデータを更新する際にも協調作業によって一貫性を維持する提案 [96, 32] や，長時間に渡りデータの更新作業を行うためにチェックアウトによりデータを取り出して更新データをチェックインすることで一貫性を維持する提案 [29, 45, 63]，あるいは，ワークフローを利用した並行作業において一貫性を確保する提案 [78] など，さまざまなデータ入力環境において一貫性を維持するための提案がなされている．しかし，大量のデータ検索を伴う一貫性の確認，例えば，実際の現金との照合用の集計や，複数のテーブル間の整合確認などは，処理効率の点からバッチ処理で実行される必要がある．従って，実際のシステム運用では，図 2.1 の仮処理でこれらの確認処理や，照合用の集計が行われ，確認用帳票により実際の現金との照合や，テーブル間の不整合の有無などが確認される．従って，データの誤りが検出された場合には該当のデータを訂正し，訂正終了後にバッチ検索処理と仮処理を再実行するという操作が，全ての確認が完了するまで繰り返される．

例えば小売システムでは各地の店舗に設置された業務端末から売上げ情報が即時に入力されて，データベースに蓄積され，さまざまな処理で利用される．これらの処理では，例えば，請求書は専用のプリプリント用紙に印刷され，口座振替データは金融機関に送付される．これらの処理で誤りがあると，誤った金額を請求したり，実際に引き落としたりしてしまい，復旧のために大きなコストが発生するだけでなく，社会や企業活動そのものに影響を与えることになる．あるいは，決算は公表されるため，決算処理では処理結果の正確性が要求される．このため，確定処理の前に仮処理が実行され，さまざまな確認作業が行われる．確認作業でデータ誤りが検出された場合には，主管部門にデータの訂正が依頼され，データ訂正後に再度，バッチ検索処理と仮処理が実行される．こうして，仮処理での確認作業が完了した後で，確定処理により業務用帳票の印刷や，口座振替データの金融機関への送信，あるいは決算データの確定が行われる．

ここで，図 2.1 に示すようにバッチ検索処理によって検索されたデータは検索結果ファイルに出力されるが，このファイルはオンライン入力の対象外である．また，仮処理および確定処理は検索結果ファイルによって実行できる．すなわち，オンライン入力とバッチ処理の並行実行に関する課題は，図 2.1 のオンライン入力とバッチ検索処理の並行実行に関する課題に帰着できる．

バッチ検索処理をオンライン入力と並行して実行するためには，検索中に行われるオンライン入力の影響を受けない検索方式であると同時に，バッチ処理で検出されたデータ誤りを訂正した上で，訂正データを反映した検索結果を繰り返し得るという操作を行うことが必要になる．また，基幹系システムのバッチ処理は，月末などの特定期日に集中することが多いため，特定期日には処理時間が長くなり，最悪の場合は所定の時間に処理が完了しないという事態が発生してしまう．これを回避するためには，処理の優先度により実行時間を分散したり，緊急性の低い処理は実行する期日を変更したりするなどの，柔軟なスケジュールを設定できることが必要になる．さらに，実際の基幹系システムに適用するためには，システム運用で発生するさまざまなデータ入力の運用や，データの管理，検索に対応できる必要がある，

すなわち，実際に基幹系システムに適用するためには，以下の要件に対応できる必要がある．

- **バッチ検索処理の要件**

要件 1 オンライン入力との並行実行
バッチ検索処理実行中にオンライン入力が行われても，その影響を受けない検索結果が得られること．

要件 2 データ訂正時の再実行
バッチ処理でデータ誤りが検出された場合には，オンライン入力中であってもその入力の影響を受けずに，データ訂正とバッチ検索処理を繰り返し実行できること．

要件 3 スケジュールの柔軟性
特定時刻のデータを利用する処理であっても，特定時刻以降の任意の時刻に該当処理を実行することにより負荷の分散が図れること．

要件 4 実際の業務システムへの適合性
実際のシステム運用で発生するデータ入力，データ管理，検索を行えること．

2.3 時制データベースの研究動向

時制データベースに関しては，さまざまな方面からの研究が行われている [4, 9, 19, 28, 43, 49]．本節では，まず，本研究の基礎技術である時制データベースの概要を説明し，次

いで，本研究の基礎となる時間属性の表現，およびデータベース実装について従来研究を概観する．

2.3.1 時制データベースの概要

データベースは対象となった実世界の事物に関する情報を管理する．ここで，実世界は静止しているわけではなく，時間の経過とともに刻々とそのありようが変わっていくため，情報システムによっては処理を行うために最新の情報だけでなくデータの変化の履歴が必要であり，さらに，データの履歴を解析することで重要な情報を得られることが多い [49]．また，データベースのデータ更新の履歴を解析することによって，対改ざん防止の効果や監査性など，コンプライアンスの点で有益な効果を得ることができる [16]．

時系列のデータ履歴を管理するため，時制データベースでは以下の３つの時間属性を定義している [33, 49]．[1]

- **有効時間（valid time）**
  ある事実が実世界で有効だった時間．有効時間により人事記録，購買記録，株価，気象データなど，時系列の変化の履歴を管理することができる．

- **トランザクション時間（transaction time）**
  ある事実がデータベースに存在していた時間であり，エンドユーザには操作を許していない．トランザクション時間により，データベースにおけるデータの追加，変更，削除の操作履歴が管理できる．

- **ユーザ定義時間（user-defined time）**
  利用者によって定義されたシステムが管理しない時間．ユーザ定義時間は，有効時間やトランザクション時間以外の時間情報が必要になった場合に，データベースのテーブルの属性の１つとして追加される．従って，ユーザ定義時間を複数持つテーブルも定義できる．

各々の時間属性の例として，人事記録における昇進では，有効時間が昇進した日付，トランザクション時間がその異動をデータベースに入力した時刻，ユーザ定義時間が昇進の承認された日付となる [33]．

[1] 時制データベースの用語は，資料によって異なる場合がある．他の用語を付録 A に示す．

表 2.1: サポートする時間属性によるデータベースの分類

データベース	有効時間	トランザクション時間
スナップショットデータベース		
有効時間データベース	●	
トランザクション時間データベース		●
バイテンポラルデータベース	●	●

なお，有効時間を扱う場合には，実世界の時刻は刻々と変化しており，検索を行う場合には検索時点現在の時刻が最新時刻になる．同じく，トランザクション時間を扱う場合にも，検索時点現在の時刻は刻々と変化する．このような検索時点現在の時刻は「now」で表現される [38]．また，時間軸の最小の単位はクロノン（chronon）と呼ばれ，クロノンの大きさである粒度（granularity）は応用によって異なる．例えば，人事管理の事例では，トランザクション時間の単位はデータベースへの追加の頻度によって決定されるため，通常のオンライン入力では 1 秒などの比較的短い時間の単位になる．一方，人事異動は一般的に日単位で発令されるため，この場合には有効時間の単位は 1 日単位に設定される．時制データベースはサポートする時間によって分類される．表 2.1 に各々のデータベースがサポートする時間属性を「●」で示す．

時制データベースの事例として，企業の社員に関する所属部門の異動履歴を，各々の時制データベースで表現した事例を示す．事例では，2001 年 4 月 1 日に入社して開発課に配属になり，2004 年 4 月 1 日に営業課，2006 年 4 月 1 日に人事課に異動になったが，データベースには人事課を誤って総務課と入力したため，2007 年 7 月 1 日に訂正された場合を示す．

最初に，図 2.2 に有効時間データベースにおける表現を示す．有効時間データベースではデータが実世界で有効であった時間は，開始時刻を v_a，終了時刻を v_d とするとき $[v_a, v_d)$ で表現される．図 2.2 の (b) では，開発課が $[2001.4.1, 2004.4.1)$，営業課が $[2004.4.1, 2006.4.1)$，人事課が $[2006.4.1, now)$ となる．有効時間データベースでは実世界の異動は表現されるが，最新のデータしか保持しないため，訂正を行うとそれ以前のデータは更新によって捨てられてしまう．従って，最新の状態 (b) では，(a) に存在した 2007 年 6 月 31 日現在の総務課への異動の情報は保持されない．

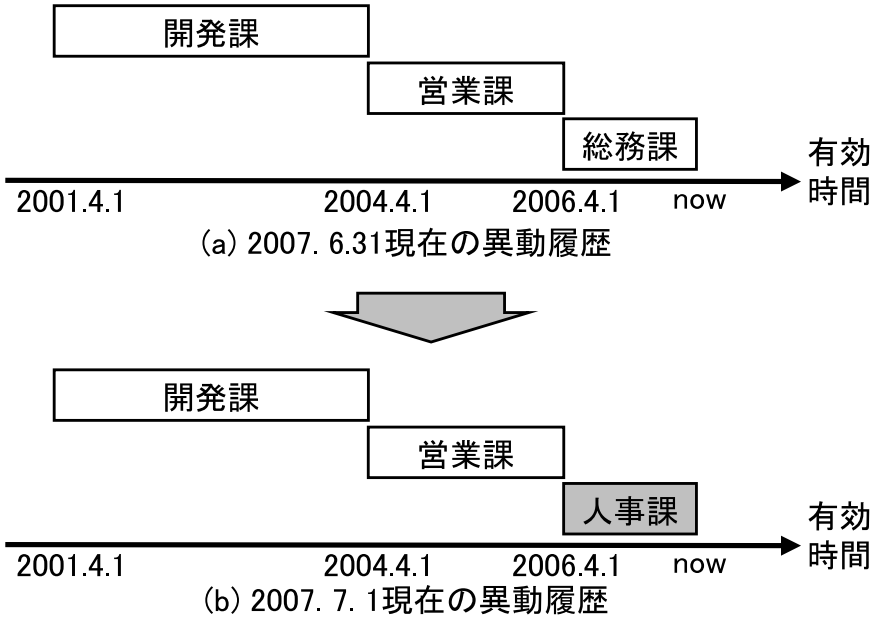


図 2.2: 人事異動の有効時間データベースにおける表現

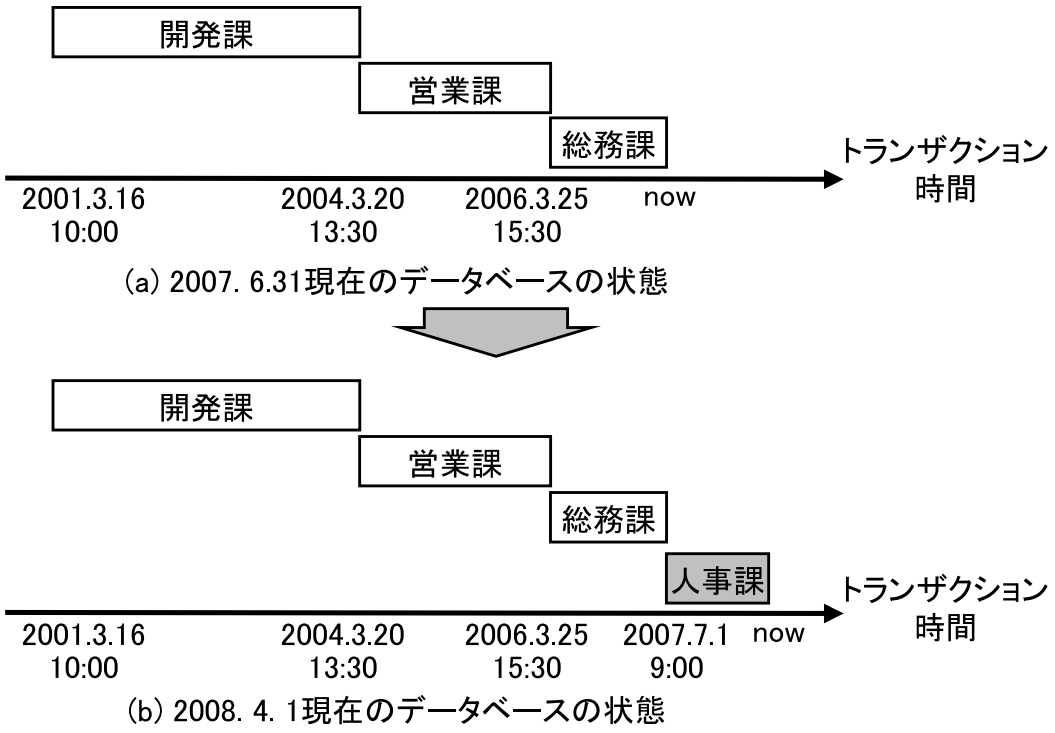


図 2.3: 人事異動のトランザクション時間データベースにおける表現

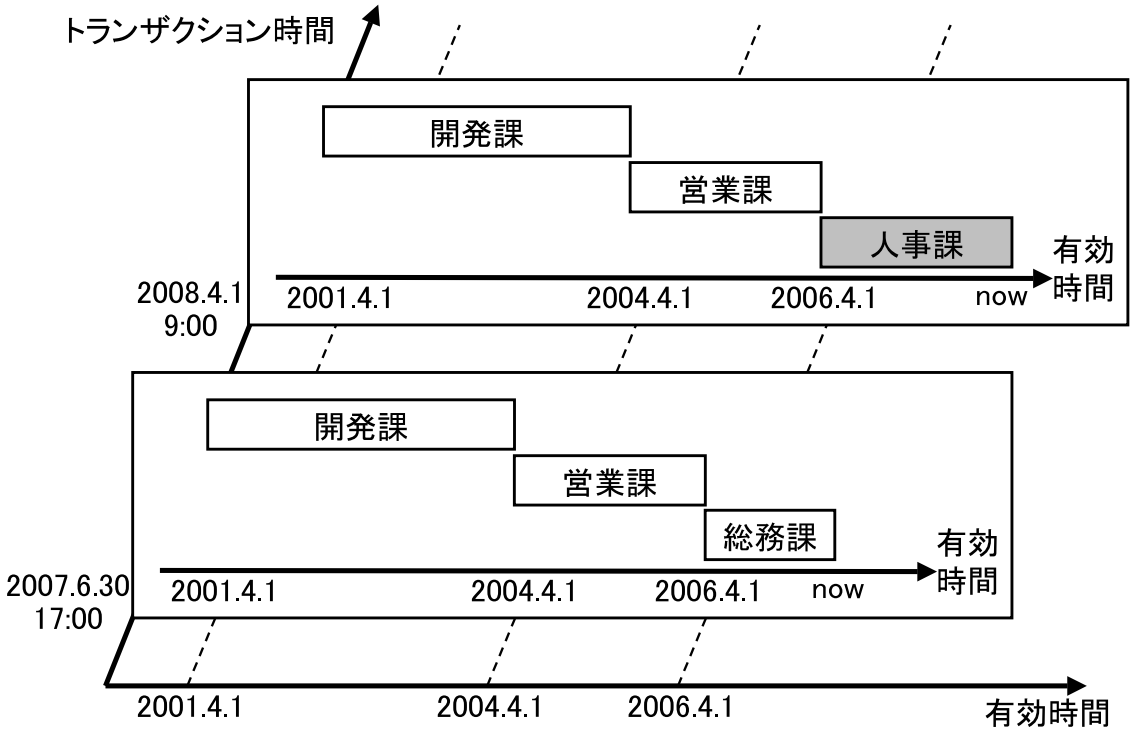


図 2.4: 人事異動のバイテンポラルデータベースにおける表現

次に，図2.3にトランザクション時間データベースにおける表現を示す．トランザクション時間データベースでは，一旦，追加されたデータを削除しても，論理的に削除されるだけで，物理的には履歴が残される．変更の場合も同様に元データが論理的に削除され，新たなデータが追加される．従って，一旦，追加されたデータはその履歴が保存されるため，訂正前の総務課への異動の履歴が残される．データがデータベースに存在した時間は，データが追加された追加時刻を t_a, 削除された削除時刻を t_d とするとき $[t_a, t_d)$ で表現される．ただし，各データの追加時刻はデータがデータベースに入力された時刻であり，実際の異動の期日とは一致しない．従って，図2.3に示されるように，例えば，最初の開発課への配属は入社の2001年4月1日ではなく，データの入力された2001年3月16日10時になる．

最後に，図2.4にバイテンポラルデータベースにおける表現を示す．有効時間とトランザクション時間は直交した時間の概念であり，バイテンポラルデータベースでは二次元の時間の表現により，データの履歴が表現される．すなわち，図2.4に示すように，あるトランザクション時間時点の有効時間の表現を得ることができ，逆にある有効時間時点のトランザクション時間の表現を得ることもできる．例えば，図2.4に示すようにトランザ

クション時間が 2007 年 6 月 30 日 17 時時点の有効時間による異動の履歴と，トランザクション時間が 2008 年 4 月 1 日 9 時時点の履歴を比較することによって，この間の訂正である，2006 年 4 月 1 日の異動の総務課から人事課へ訂正を把握することができる．

なお，近年では時間と空間の管理を統合した，時空間概念データモデルの研究や [48, 92, 24]，時間に関する推論の研究 [7] も進められている．

2.3.2 時間属性に関する検索の表現

時制データベースでは，時間に関する検索が重要になる．古典的なリレーショナルデータベースでは，時間に関するデータ型や，これらのデータ型に関する演算は充分に備わってはいなかった．このため，時制データベースの研究と並行して，時制データベースに関する質問言語の研究が行われてきた [6, 9, 11, 15, 20, 21, 30]．例えば，Snodgrass らにより TQuel（Temporal Query Language）[34, 101] の設計が行われた．この言語では，時間の履歴を時間区間（time interval）として表現しており，時間に関する演算が定義されている．TQuel の事例を図 2.5 に示す．この事例は，学部の教官のリレーション [100]$Faculity(Name, Rank)$ と助教授のリレーション $Associates(Name)$ について，Tom が助教授に昇進したとき教授であった人の名前を検索する TQuel の検索文になっている [101]．検索文に示されるように，時間の扱う単項演算子「begin of」や，2 項演算子「overlap」が定義され，時間の演算を行えるようになっている．ここで，begin of は開始時点，overlap は 2 件の時間区間の重なりを表現する．

時間区間については，Allen によって提案された時間区間論理（interval-based temporal logic）の研究 [47, 101] がある．時間区間論理では，時間区間を時間軸上において開始時刻（stp; start time point）と終了時刻（etp; end time point）をもつ閉区間 $[stp, etp]$ としたとき，時間区間の間に図 2.6 に示す 13 種類の 2 項の時間的関係（temporal relation）が定義できることを示している．図では，左に時間区間の時間的関係の表現，右にその意味を示す．

これらの時間属性に関する概念に基づき，SQL の拡張としての言語の提案が行われた [21, 30]．特に，SQL2 の拡張として提案された TSQL は標準規格 SQL99 への取り込みが提案され [20, 36, 76, 98, 101]，商用データベースでも時間属性が扱えるようになっている [27, 40]．このような時間属性に関しては，応用面における評価の報告として，時間以外の属性が必ずしも一致していないテーブル同士を合成し時間順に並べるという演算での有用性が指摘されている [22]．また，近年ではオブジェクト指向データベース [56, 94]

```
/* Faculty (Name, Rank)  教官の名前とランク */
range of f is Faculty
/* Associate (Name)  助教授の名前 */
range of a is Associates
/* 教授の名前を質問 */
retrieve into Full (Name=f.Name)
  /* Tomが助教授( a ) で教授( f )だった教官 */
  where a.Name = “Tom” and f.Rank = “full”
  /* 時期は助教授( a )の開始時点 */
  when f overlap begin of a
```

図 2.5: TQuel による検索構文の事例

1) equal (m1, m2)	iff m1.stp=m2.stp and m1.etp=m2.etp
2) before (m1, m2)	iff m1.etp<m2.stp
3) after (m1, m2)	iff before (m2, m1)
4) during (m1, m2)	iff m2.stp<m1.stp and m1.etp<m2.etp
5) contains (m1, m2)	iff during (m2, m1)
6) overlaps (m1, m2)	iff m1.stp<m2.stp<m1.etp and m2.stp<m1.etp<m2.etp
7) overlapped by (m1, m2)	iff overlaps (m2, m1)
8) meets (m1, m2)	iff m1.etp=m2.stp
9) met_by (m1, m2)	iff meets (m2, m1)
10) starts (m1, m2)	iff m1.stp=m2.stp
11) started_by (m1, m2)	iff strats (m2, m1)
12) finishes (m1, m2)	iff m1.etp=m2.etp
13) finished_by (m1, m2)	iff finishes (m2, m1)

図 2.6: Alle による時間区間の 13 種の時間的関係

人事異動テーブル			
社員番号	所属部門	有効時間(開始)	有効時間(終了)
001	開発課	2001.4.1	2004.4.1
001	営業課	2004.4.1	2006.4.1
001	人事課	2006.4.1	now

図 2.7: 人事異動のリレーショナルデータベースでの実装事例

の概念が標準規格の SQL および商用データベースに取り入れられた結果 [2, 27, 51, 108]，ユーザが時間属性や演算を定義し，これを利用することが可能になっている．これらの標準化によって，データベースの検索における時間属性の扱いが容易になってきている．

2.3.3 時制データベースの実装

■ 時間属性の実装および応用

有効時間は，時間の経過に伴って変化する実世界の状態の履歴を表現するために使用される．業務システムのデータベースでは，有効時間による実世界の履歴管理が必要になる場合が多い．例えば，人事管理では，職歴，研修履歴，昇給・昇格や給与などの履歴が管理され，人事異動や昇給時期，退職金の計算などは該当者の過去からの履歴を参照して決定される．また，会計システムで決算を行うためには，入金や出金の期日を管理して決算の該当期間のデータを抽出できることが必要になる．さらに，データの履歴解析としては，時系列データのデータマイニングにより過去に蓄積された膨大なデータから規則性を抽出して，なぜそれが起こっているのかの理解や，今後の予測を行うために利用される事例がある [94]．例えば，小売業では，購入履歴の分析によりある特定時点に起きた現象の原因の分析や，ある商品を購入した顧客は次になにを購入するかという購買パターンの分析に利用されている．また，株価や気象変動などでは，過去のデータを分析し，将来を予測するために利用される．

有効時間はユーザによって入力，管理されるため，データベースでは通常の業務データ

の属性として実装できる．例えば，図 2.2 の (b) における人事異動の有効時間データベースにおける表現をリレーショナルデータベースに実装する場合には，図 2.7 に示すように各々の人事異動を表現したデータの集合により履歴が表現される．図 2.7 では，社員番号 = 001 の社員とし，該当の所属部門の配属日を「有効時間（開始）」で，他の部門への異動日を「有効時間（終了）」で示している．従って，SQL により，例えば 2004 年 5 月 15 日の所属部門は以下で検索できる．

$$select\ \text{所属部門}\ from\ \text{人事異動テーブル}\ where\ \text{社員番号} = '001'\ and$$
$$\text{有効時間（開始）} \leq 2004-05-15\ and\ 2004-05-15 < \text{有効時間（終了）} \quad (2.1)$$

なお，ユーザ定義時間も有効時間と同様にユーザによって入力，管理されるため，有効時間と同様に実装することができる．

これに対し，トランザクション時間は，ある事実がシステム内に存在していた時間であるため，システムにより管理され，ユーザには入力や変更手段が公開されない．業務システムでは不正な操作により多額の損害が発生することがある [68]．従って，このような不正に対しデータの改ざんを検知したり，監査によって改ざんなどの不正がないことを確認したりするための，データの経緯の管理が重要になる．このような応用事例として，トランザクション時間によりデータの入力，削除時刻を管理し，一旦，追加されたデータは物理的に削除されない「大福帳システム」データ管理による会計システムが実現され，実際の基幹系システムに適用されている [68].

また，データベースの履歴により任意の時刻のスナップショットを検索できる．これを利用して，対象データの変更履歴の検索，時系列の比較，差分の抽出，推移分析などが行われる他，データベースをある時点の状態に復元するためにも応用されている [16]．例えば，この機能を利用することにより，データに対する誤った更新，削除などの操作が発生した場合にも，元のデータを検索して確認したり，該当データを操作前の状態に復元したりすることが可能になる．

トランザクション時間データベースはリレーショナルデータベース上での実装の研究が行われている [18, 35]．業務システムでの実装にあたっては，ユーザがトランザクション時間やデータそのものを物理的に操作できない構成にする必要がある．すなわち，ユーザのデータベース更新手段を制限し，データの追加，変更，削除にあたってはトランザクション時間がシステムにより設定され，一旦追加されたデータは更新や削除した後も検索可能なように，論理的な削除のみを行う構成にする必要がある．なお，リレーショナ

(a) テーブルのデータ

人事異動テーブル					
社員	所属	Ta	Td	Va	Vd
001	202	2001	2004	2001	now
001	202	2004	now	2001	2004
001	303	2004	now	2004	now

部門テーブル					
所属	名称	Ta	Td	Va	Vd
202	研究課	1990	2000	1990	2000
202	研究課	2000	now	1990	2000
202	開発課	2000	now	2000	now
303	営業課	1990	now	1990	now

(b) 単純な結合演算の結果

人事異動テーブル						部門テーブル				
社員	所属	Ta	Td	Va	Vd	名称	Ta	Td	Va	Vd
001	202	2001	2004	2001	now	研究課	1990	2000	1990	2000
001	202	2001	2004	2001	now	研究課	2000	now	1990	2000
001	202	2001	2004	2001	now	開発課	2000	now	2000	now
001	202	2004	now	2001	2004	研究課	1990	2000	1990	2000
001	202	2004	now	2001	2004	研究課	2000	now	1990	2000
001	202	2004	now	2001	2004	開発課	2000	now	2000	now

図 2.8: 人事異動のバイテンポラルデータベースでの実装事例

ルデータベースでのデータの構成自体は有効時間と同様であり，式（2.1）に示す有効時間と同様の検索が可能になる．さらに，最近ではトランザクション時間データベースのDBMS として，Herodotus[16] が商用化されている．Herodotus は WORM（Write Once Read Many）の概念に基づいた追記型のデータベースで，SQL/92 に対応するとともに，履歴ダイアログを用いて容易に過去の任意時点のデータ履歴にアクセスできる機能を備えている．また，データの更新履歴の操作を一般のユーザに非公開にし，システム管理者のみにデータの履歴管理を許すことで，業務システムの操作に関する内部牽制機能の実現や，耐改ざん性，監査性を維持して不正を防止する仕組みが構築できる．

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を管理する．従って，業務システムでの実装にあたっては，トランザクション時間データベースと同様に，トランザクション時間をシステムで管理するという機能が必要になる．バイテンポラルデータベースも，リレーショナルデータベース上での実装が研究されている[11, 35]．ただし，業務システムへの適用にあたっては，有効時間とトランザクション時間の双方の履歴を管理する必要があるため，データ量の増大や検索の複雑化という課題を解決する必要がある．図 2.8 に，図 2.4 に示したバイテンポラルデータベースにおける人事異動のうち，開発課から営業課への異動を「人事異動テーブル」に示す．図 2.8 の例では，部門名称を「部門テーブル」に持たせる構成とし，2000 年に開発課の部門名称が変

更された事例を示す．図で，T_a はトランザクション時間の追加時刻，T_d は同じく削除時刻，V_a は図 2.7 と同様に有効時間の開始時刻，V_d は同じく終了時刻を示す．また，簡単のため各時刻は年のみを示している．

まず，データ量の増大については，トランザクション時間データベース，あるいは有効時間データベースにおける履歴の追加が 1 件のデータの追加であるのに対し，バイテンポラルデータベースでは図 2.8 の (a) の人事異動テーブルに示すように，2004 年の 所属 = 202 から 所属 = 303 の異動 1 回で 2 件の履歴データが追加される．これは，トランザクション時間による履歴管理のため，まず，所属 = 202 について，$V_d = now$ の履歴を論理的に削除し，新たに $V_d = 2004$ のデータを追加する必要があること，さらに新たな履歴である 所属 = 203 のデータが追加されることによる．また，部門名称のテーブルでは，2000 年に部門名称が「研究課」から「開発課」に変更になったことに伴い，同じく 2 件の履歴が追加される．このように，バイテンポラルデータベースでは履歴によるデータ量が増大し，大量のデータを扱うシステムでは履歴のデータ量の増大が課題となる．

また，検索処理の複雑化については，図 2.8 の (b) に，人事異動テーブルと部門テーブルの「所属」による結合演算の結果のうち，所属の条件が 所属 = 202 のものを示す．スナップショットデータベースであれば，各々の該当レコードが 1 件であるため，結果のデータも 1 件となる．しかし，バイテンポラルデータベースでは履歴があるため，検索結果は双方の直積となり，6 件の結果が得られてしまう．ここで，トランザクション時間データベース，あるいは有効時間データベースであれば，人事異動テーブルの 所属 = 202 のデータは 1 件，部門テーブルのデータは 2 件であり，結果は 2 件に留まるため，検索結果から対象データを抽出するという操作が可能になる．しかし，バイテンポラルデータベースで，大量のデータを検索する場合には，検索の段階でデータの絞込みを行う必要があり，検索処理が複雑化するという課題がある．

このように，バイテンポラルデータベースで大量のデータを扱う場合にはデータ量の増大，検索処理の複雑化が課題になる．このため，実際の基幹系システムに適用，評価した事例は見当たらない．また，DBMS として商用化された事例も，ほとんどないことが指摘されている [11, 49].

■ 時間属性による同時実行制御の実装

時制データベースに関しても，スナップショットデータベースと同様に一貫性の維持が重要であり，研究が行われている [10, 13, 26, 39]．ここで，時刻を利用した同時実行制御

表 2.2: 楽観的方法の確認フェーズでの判定

	データのトランザクション時間	
データのロック	更新されている	更新されていない
ロック中	他のトランザクションが更新中	
ロックなし	他トランザクションが更新済	他トランザクションの更新なし

には時刻印方式 [71, 94] や楽観的方法 [25, 99] がある．このうち，楽観的方法はトランザクション時間を利用した方式であり，トランザクションはオブジェクトを操作する前にロックなどの制御を行わず，トランザクションが終了するときに他のトランザクションと競合があったかを調べるため，データベースのロック時間を短くできるという利点がある．楽観的方法については，さまざまな実装方式が提案されている．以下に例を示す．この例では，各フェーズで同時実行制御のために以下処理が行われる．

- **読み込みフェーズ**
  トランザクションは，更新対象データをデータベースから読み出す．このときは，データベースに対するロック制御は行わない．読み出したデータはトランザクション内に保存され，処理されて更新用データが作成される．

- **確認フェーズ**
  更新対象データを更新モードでロックして再度読み込み，表 2.2 に示す条件で，他トランザクションの更新有無の判定を行う．更新対象データがロックされているか，読み込みフェーズで読み込んだデータと比較してトランザクション時間が更新されている場合は，読み込みフェーズ以降に該当データが他のトランザクションで更新されているためロールバックする．それ以外の場合には，更新フェーズを実行する．

- **更新フェーズ**
  更新用データでデータベースの更新を行う．

他のトランザクションによる更新とのタイミングを図 2.9 に示す．図で「●」は更新トランザクションの確認フェーズ開始時点における，他のトランザクションの状態を示す．読み込みフェーズの間に他のトランザクションで更新が行われた場合には，トランザクション時間が更新されて更新済の状態になる．また，確認フェーズのときに，他のトランザクションが確認フェーズや更新フェーズを実行していると，更新対象データはロック中

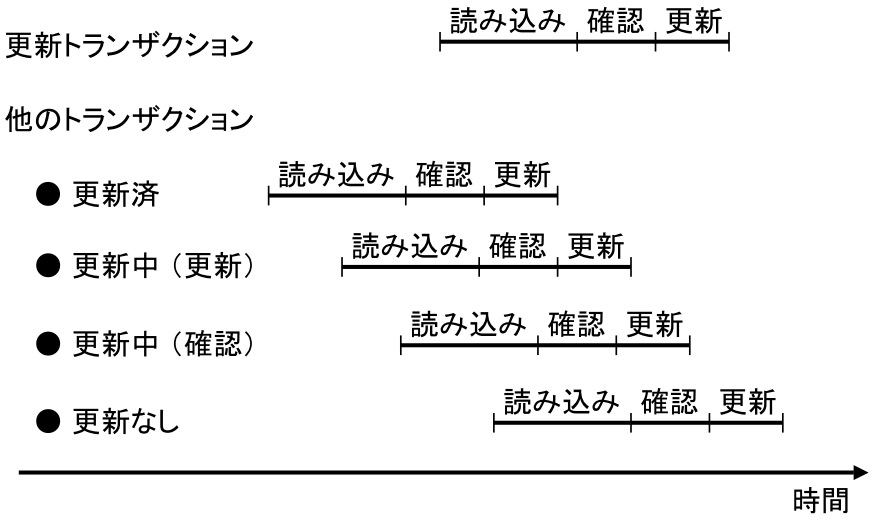


図 2.9: 楽観的方法におけるトランザクション競合のタイミング

の状態になる．確認フェーズのときに，他のトランザクションが読み込みフェーズを実行していても，更新されておらずロックなしの状態になる．従って，表 2.2 の「他トランザクションの更新なし」，すなわち図 2.9 の「更新なし」の場合のみ，更新トランザクションは更新フェーズを実行する．

業務システムのオンライン入力に適用した場合には，読み込みフェーズが画面に更新前データを表示し，変更入力を実行している段階になる．データの変更完了後にデータベースの更新指示が入力され，確認フェーズに移るが，他のトランザクションで更新済あるいは更新中の場合には再度，データの読み込みが行われ，画面にメッセージが表示されて変更入力のやり直しが指示される．それ以外の場合には，データベースが更新される．従って，楽観的方法は読み込みフェーズの処理時間が長く，かつ同一のデータを複数のトランザクションから更新する可能性の低い業務システムに適した方法といえる．

■ 検索処理の実装

時制データベースでは時系列のデータの履歴検索が可能である反面，データベース内に蓄積されるデータは常に増え続けるという特徴がある．例えば，トランザクション時間に関しては，一旦，追加されたデータの更新や削除は論理的な操作として行われ，物理的な更新や削除は行われない．従って，データの変化や，更新の履歴が永久に保存されるという特徴を持つ．このため，時制データを高速に検索するための技術が提案されている．

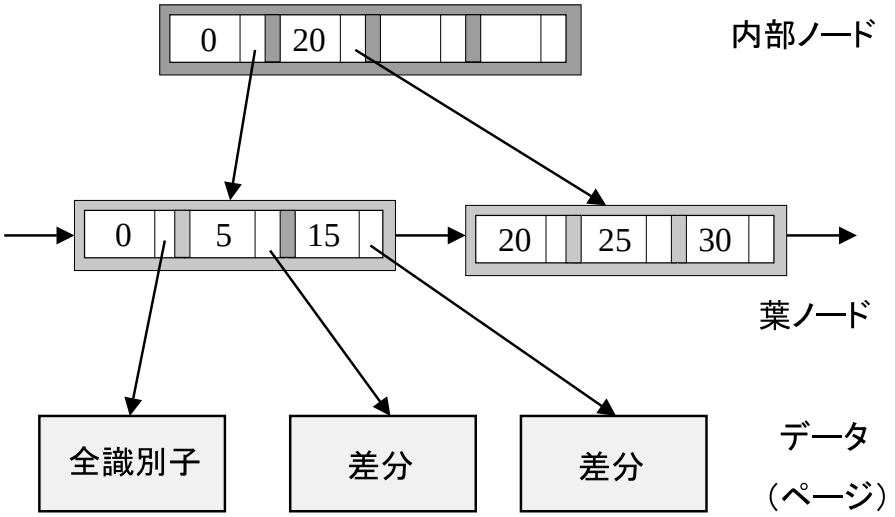


図 2.10: タイムインデックス

例えば，時制データを高速に検索するための索引技術やアルゴリズム [14, 49]，トランザクション時間データベースを対象とした高速化の提案 [3, 23, 41] が行われている．

このうち，バイテンポラルデータベースに対応できる検索方式として，図 2.10 に示す，タイムインデックス（time index）が提案されている [49]．この方式は，B+-tree をベースにした方式であり，ノードのインデックスは単調増加の時刻になる．また，索引時点（indexing point）の概念が導入されている．これは，何らかの変化の発生した時間であり，例えば，トランザクション時間の追加時刻が t_a，削除時刻が t_d であるデータがある場合には，t_a，t_d が索引時点となる．ここで，検索を効率的に行うためには全ての時刻におけるスナップショットのコピーを保持するのが理想であるが，空間効率が悪く現実的でない．このため，各葉ノードの先頭エントリから示されるページは，その時点で存在した全てのデータの識別子を保持し，続くエントリは索引時点に基づく差分が保持される．

この方式は，トランザクション時間と有効時間の双方を索引として使用でき，終了時刻が now の時間区間を扱うことができる．また，ある時点の検索と，時間区間の検索の双方を高速化できるという特徴がある [49]．

2.4 バッチ検索処理に対する従来技術の課題

本節では，時制データベースに関する従来技術によってバッチ検索処理を行う場合の課題を示す．まず，トランザクション時間データベースによりバッチ検索処理を構成する方法を示し．次にバッチ検索処理の要件に対する課題を示す．さらに，有効時間データベース，バイテンポラルデータベースの課題を示す．

2.4.1 トランザクション時間データベースによるバッチ検索処理

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のスナップショットを検索でき，また，過去のスナップショットは現在のオンライン入力の影響を受けない．あるデータがデータベース内で有効だった時間区間を $[t_a, t_d)$ で示す．ここで，t_a，t_d は 2.3.1 節に示す追加時刻，削除時刻であり，データが削除されていない場合には $t_d = now$ で表現される．データの変更を行う場合には，変更前のデータに削除時刻を設定して論理的に削除し，変更後のデータを追加する．

この様に，一度追加されたデータは物理的に削除されずに残されることから，トランザクション時間 t を指定して $t_a \leq t < t_d$ なるデータを検索すれば時刻 t 現在のスナップショットが表現され，オンライン入力中であってもデータの一貫性が保たれる．図 2.11 に，バッチ検索処理によって，オンライン入力中に時刻 $t = t_q$ のスナップショットを検索する例を示す．図の (a) の now の時点が検索開始時刻であり，図の (b) の now の時点が検索完了時刻になる．検索中に，$ID = D2$ の変更および $ID = D3$ の追加が行われた．検索開始時刻に変更済のデータ $ID = D1$ は検索前後で変化せず，検索中に追加されたデータ $ID = D3$ は検索対象にならない．検索中に変更されたデータ $ID = D2$ の変化を図 2.12 に示す．変更された時刻 t_d に対し，スナップショットの時刻 t_q は $t_q < t_d$ となるため，変更前のデータが検索対象となり，オンライン入力の影響を受けない．

従って，検索時間が長時間におよぶバッチ検索処理でもオンライン入力の影響を受けず，過去の任意の時刻のスナップショットを検索できることから，バッチ検索処理の要件 1 と要件 3 を満たすことができる．

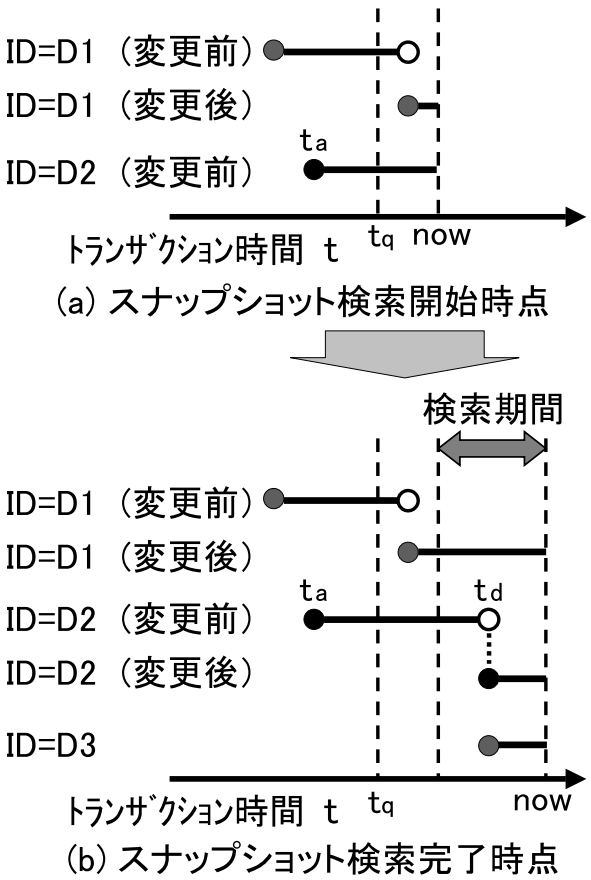


図 2.11: オンライン入力中のスナップショット検索

(a) スナップショット検索開始前

ID	追加時刻	削除時刻	値
D2	t_a	now	1,000

(b) スナップショット検索完了後($t_q < t_d$)

ID	追加時刻	削除時刻	値
D2	t_a	t_d	1,000
D2	t_d	now	2,000

図 2.12: 検索中に変更されるデータの動き

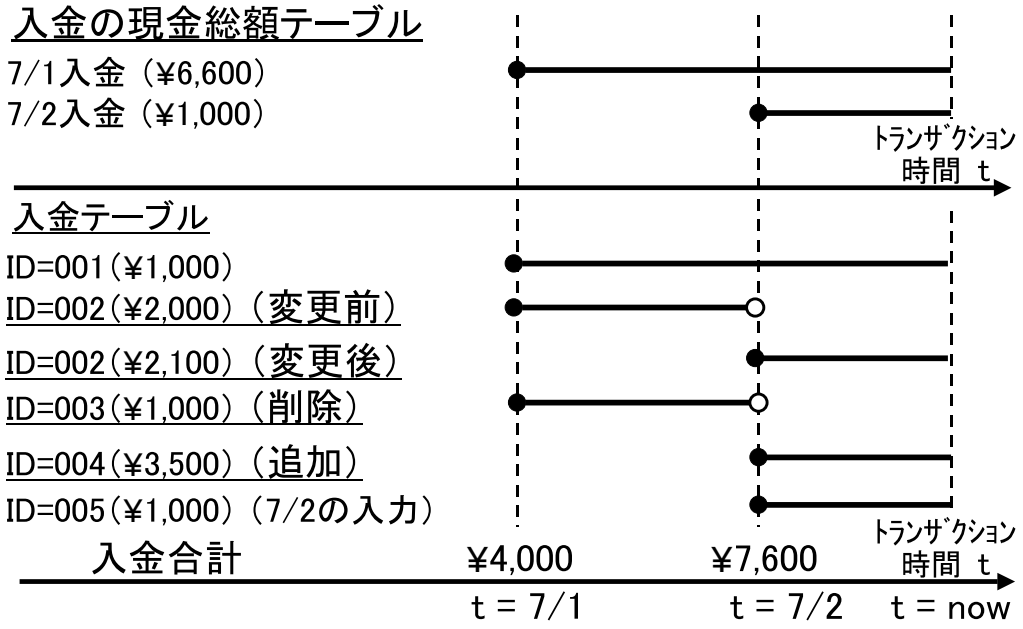


図 2.13: トランザクション時間データベースのデータ訂正時の再実行に関する課題

2.4.2 トランザクション時間データベースの課題

スナップショットによる検索を，小売システムに適用した例を図 2.13 に示す．なお，以下の例も含めてトランザクション時間は日単位としている．実際の実装においては，データ更新頻度などのシステム要件に応じて時間の単位が決定される．図で，7 月 1 日の入金テーブルを入金の現金総額テーブルと突合せた結果，$ID = 002$，$ID = 003$，$ID = 004$ で各々，入力誤り，重複入力，入力漏れが検出され，7 月 2 日に変更，削除，追加による訂正が行われた．また，7 月 2 日に新たに $ID = 005$ の入金データが追加されている．7 月 1 日のスナップショットでは，これらの 7 月 2 日の更新は反映されない．この特性により，オンライン入力によるデータ更新中であっても，長時間のデータベース検索において検索結果の一貫性を保つことができる．

ここで，実際の業務においては 7 月 1 日の正しい集計結果，すなわち現金総額テーブルと整合した検索結果である，6,600 円が得られることが必要になる．しかし，図 2.13 の例では，7 月 1 日のスナップショットでは訂正前のデータが検索され，7 月 2 日では 7 月 2 日の入金分である $ID = 005$ のデータが検索対象になるため，7 月 1 日の訂正された状態は検索できない．すなわち，バッチ検索処理の要件 2 が満足できないという課題がある．

このデータ訂正を反映した再検索が行えないという問題はテーブル間の整合の訂正だけでなく，異動統計などの定期的に行われる処理においても課題になる．例えば，小売シ

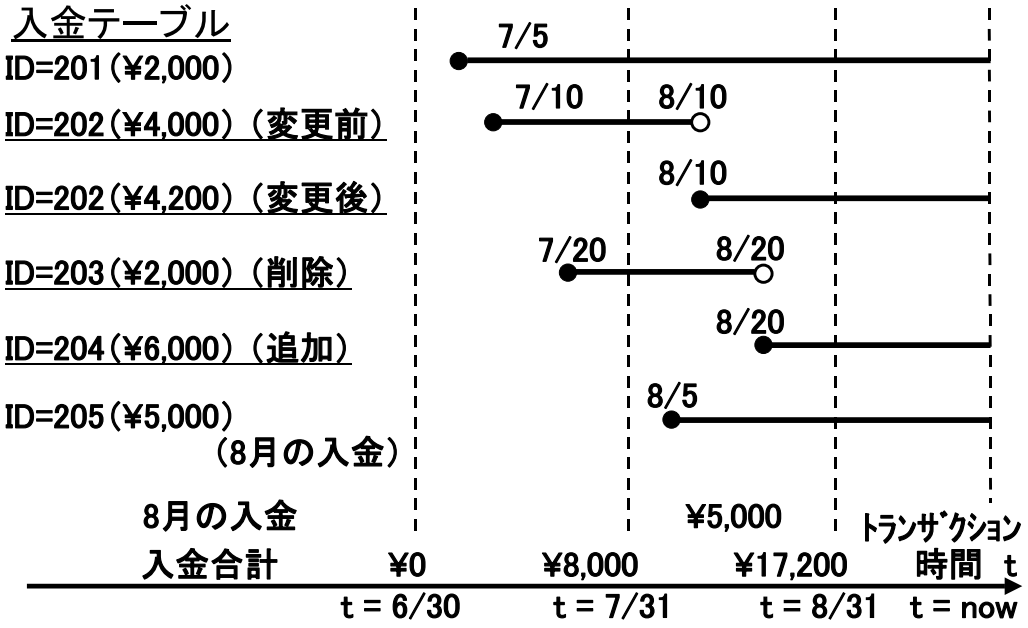


図 2.14: トランザクション時間データベースのスケジュールの柔軟性に関する課題

ステムでは入金データの推移を管理時するため，時系列の集計，すなわち日次，月次などの定期的な集計が行われる．ここで，各集計の差引きと，この間の入金は一致する必要がある．トランザクション時間データベースでは過去のトランザクション時間のスナップショットが検索可能であり，今回の集計を行う時点で，前回の集計および前回の集計以降の入金を同時に検索できる．しかし，前回の集計以降にデータの訂正が行われた場合には，前回の集計に訂正が反映されないため，前回と今回の集計の差引きと，この間の入金に不整合が発生するという課題がある．

図 2.14 に入金データの訂正を行った場合の例を示す．7 月の入金 $ID = 202$，$ID = 203$，$ID = 204$ で各々，金額誤り，重複入力，入力漏れが検出され，8 月に変更，削除，追加による訂正が行われた．また，8 月に新たに $ID = 205$ の入金データが追加された．7 月入金のデータを訂正した結果，8 月 31 日現在の入金合計 17,200 円と 7 月 31 日現在の入金合計 8,000 円の差引き 9,200 円に対し，8 月の入金は 5,000 円で整合していない．この場合にも，$ID = 202$，$ID = 203$，$ID = 204$ の訂正結果を 7 月 31 日現在の検索結果に反映する必要があるが，訂正完了時点の 8 月 20 日を指定した検索では，$ID = 205$ も検索対象になってしまう．すなわち，複数の期日の処理を後日一括して実行すると，この間にデータの訂正が発生した場合には，各期日のデータの間に矛盾が発生するという課題がある．

このように，トランザクション時間データベースでは，バッチ検索処理の要件 2 に対して課題がある．

2.4.3　その他の時制データベースの課題

有効時間データベースでは実世界の履歴を管理するが，トランザクション時間を管理していないため，2.4.1 節に示した方式が使用できない．従って，有効時間データベースではスナップショットデータベースと同様の検索を行う必要がある．すなわち，1.2 節に示したスナップショットデータベースによるバッチ検索処理と同様の課題がある．

バイテンポラルデータベースでは，トランザクション時間を管理しているため，トランザクション時間データベースと同様に，2.4.1 節に示した検索を行うことができる．さらに，トランザクション時間として t_1 を，有効時間として t_2 を指定して検索を行うことにより，実世界の時刻 t_2 の状態に関して時刻 t_1 までに行われた訂正を反映したスナップショットを検索することができる．この検索方法の詳細については，第 3 章に示す．しかし，バイテンポラルデータベースは 2.3.3 節に示したように実装に関する課題があり，基幹系システムの実際のシステム運用に適用，評価した事例はない．すなわち，バッチ検索処理の要件 4 の実際の業務システムへの適合性が検証されていないという課題がある．

2.5　むすび

本章では，まず，オンライン入力と並行してバッチ検索処理を行うための要件を整理し，オンライン入力との並行実行，データ訂正時の再実行，スケジュールの柔軟性，実際の業務システムへの適合性が必要であることを示した．

また，時制データベースの研究動向，特に本研究の基礎となる時間属性の表現と，データベースの実装について概観した．さらに，トランザクション時間データベースでは，バッチ検索処理の要件のうち，データ訂正時の再実行に関する課題があること，有効時間データベースはスナップショットデータベースと同様の課題があること，バイテンポラルデータベースは実装に関する課題があり，実際の業務システムへの適合性が検証できていないことを示した．

このように，時制データベースによってオンライン入力と並行して実行できるバッチ検索処理を構築するためには，トランザクション時間データベースでは方式の改良が必要で

あり，バイテンポラルデータベースでは実際の基幹系システムでの実証が必要である．

第3章　バイテンポラルデータベースによる基幹系システムの構築

3.1　まえがき

本章では，バイテンポラルデータベースにより 2.2 節に示したバッチ検索処理の要件のうち，オンライン入力との並行実行（要件 1），データ訂正時の再実行（要件 2），スケジュールの柔軟性（要件 3）を満たす検索方式が構成できることを示す．また，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した結果を示し，これを，実際の業務システムへ適合性（要件 4）の点から評価，考察する．

バイテンポラルデータベースでは，トランザクション時間を管理しており，2.4.1 節に示したトランザクション時間データベースの検索方式によってオンライン入力との並行実行，およびスケジュールの柔軟性を備えた検索が可能である．さらに，有効時間を管理しているため，トランザクション時間と有効時間の両方を指定してスナップショットを検索できる．以下で，トランザクション時間を t_1，有効時間 t_2 で示す．ここで，トランザクション時間 t_1 と有効時間 t_2 を指定した検索は，実世界の時刻 t_2 の状態に関して時刻 t_1 までにシステムに入力された訂正を反映したスナップショットになる．すなわち，データ訂正が発生した場合には，t_1 として訂正後の時刻を指定して検索処理を再実行することにより，訂正を反映した結果を得ることができる．本章では，まず，バイテンポラルデータベースによるバッチ検索処理を，バッチ検索処理の要件 1，要件 2，および要件 3 の点から評価し，これらの要件を満たす検索ができることを示す．

また，これを基幹系システムに適用するためには，実際のシステム運用で発生するさま

ざまな訂正や，データ入力，データ管理，検索などに対応できること，すなわち，実際の業務システムへの適合性（要件 4）を満たすことが必要になる．しかし，バイテンポラルデータベースを実際に基幹系システムに適用し，その運用を評価した事例は見当たらない．そこで，本章で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である自治体システムに適用した結果を示し，その結果について以下の 3 点から評価する．

第 1 に，バッチ検索処理の要件 1，要件 2，および要件 3 に対する評価，すなわちオンライン入力とバッチ検索処理の並行実行に関する評価を示す．自治体システムでは，住民からの届出や申告に基づくさまざまなデータがオンライン入力され，請求に応じて入力を反映した証明書を即時に発行する．従って，窓口業務の時間帯にはオンライン入力を停止できない．また，入力されたデータにより，定期，あるいは随時のバッチ検索処理によって統計資料の作成などが行われる．このようなシステムにバイテンポラルデータベースを適用することで，夜間バッチを削減できる効果があることを示す．

第 2 に，実際の業務システムへの適合性（要件 4）の評価として，バイテンポラルデータベースにより，実際の基幹系システムで必要な機能を構成できることを示す．さらに，長期間に渡るデータ訂正を行う業務でもデータの履歴管理が容易になること，バイテンポラルデータベースの履歴表現により内部処理での訂正と業務処理での訂正を分けて管理できること，複数の時刻のデータを統合した処理が実行できること，によるデータ管理における効果を示す．

最後に，バイテンポラルデータベースの実装の評価を示す．バイテンポラルデータベースの実装に関しては，履歴管理に伴う検索処理の複雑化とデータ量の増大という課題があった．適用システムでは，バッチ処理のワークファイルをデータベースのテーブルで構築し，有効時間のイベント表現による履歴データの削減を行った．この対策により，商用のリレーショナルデータベース上に基幹系システムに適用できるバイテンポラルデータベースを構築できることを示す．

本章は以下のように構成される．3.2 節でバイテンポラルデータベースによってトランザクション時間データベースの課題が解決できることを示し，3.3 節で基幹系システムである自治体システムに対するバイテンポラルデータベースの実装事例と，そのシステム運用について述べる．3.4 節で実装および運用の結果について評価し，3.5 節で考察する．最後に 3.6 節でまとめと課題を述べる．

3.2 バイテンポラルデータベースによるバッチ検索処理

2.4.2 節に示したトランザクション時間データベースの課題，すなわち 2.2 節に示したバッチ検索処理の要件の，データ訂正時の再実行（要件 2）に関する課題がバイテンポラルデータベースにより解決できることを示す.

3.2.1 バイテンポラルデータベースの構成

バイテンポラルデータベースのテーブルのリレーションは以下で表現される.

$$R(K, T, V, A) \tag{3.1}$$

各々の属性を以下に示す.

- $K = \{K_1, \cdots, K_m\}$
  トランザクション時間，有効時間を指定したスナップショットで生成されるテーブルの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間区間属性であり，システムにより自動生成され，ユーザには公開されない．ここで，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，2.3.1 節に示した検索時点現在の時刻 now で表現される．データの変更を行う場合には，変更前のデータに削除時刻を設定して論理的に削除し，変更後のデータを追加する.

- $V = \{V_a, V_d\}$
  有効時間の時間区間属性であり，V_a は該当データが実世界に存在した開始時刻，V_d は終了時刻を示し，検索時点でデータが有効な場合にはトランザクション時間と同様に now で表現される．有効時間に関する履歴がデータベース内に残されることから，トランザクション時間による検索の場合と同様に，指定した有効時間現在のスナップショットが表現できる.

- $A = \{A_1, \cdots, A_n\}$
  その他の属性集合を示す．なお，ユーザ定義時間の属性は，この属性集合に含まれる.

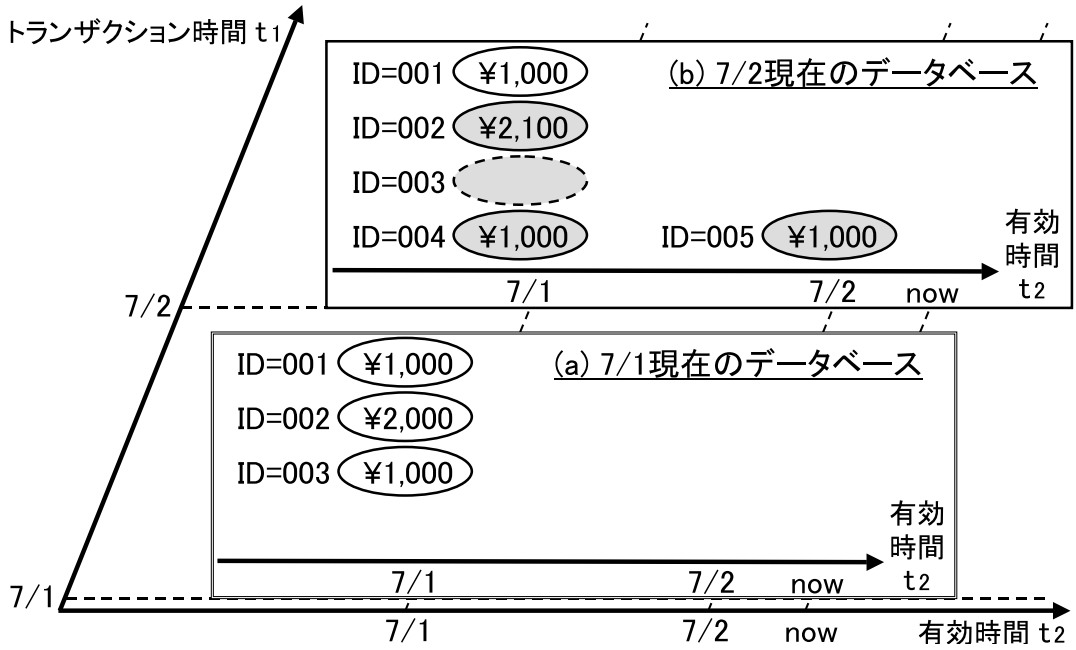


図 3.1: トランザクション時間を指定した入金テーブルのスナップショット

このように，バイテンポラルデータベースではトランザクション時間と有効時間を管理しており，データは有効時間によって表現される実世界の履歴と，トランザクション時間によって表現されるデータベースにおける履歴の，両方の時間軸の履歴として表現できる．

3.2.2 バイテンポラルデータベースのスナップショット

バイテンポラルデータベースでは，リレーション R のトランザクション時間 t_1，有効時間 t_2 の一方，あるいは両方を指定することによりスナップショットの検索を行うことができる．図 2.13 にトランザクション時間データベースの事例として入金テーブルのスナップショットを示した．これを，バイテンポラルデータベースで実現した場合の例として，以下の 3 種類のスナップショットによる検索を示す．なお，この例では，有効時間は入金日であり，V_a=V_d となる．

最初に，トランザクション時間 t_1 のみを指定したスナップショットを示す．以下で，$r[T_a]$，$r[T_d]$，$r[V_a]$，$r[V_d]$ は R に含まれるデータ r の T_a，T_d，V_a，V_d の属性値を示す [100]．このスナップショットは，トランザクション時間 t_1 を含む時間区間 $T = \{T_a, T_d\}$ を持つデー

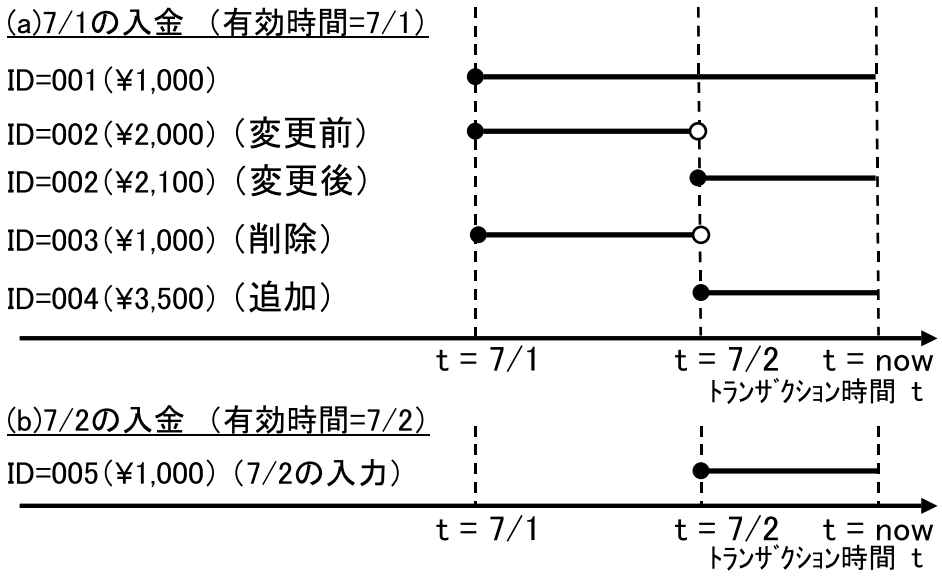


図 3.2: 有効時間を指定した入金テーブルのスナップショット

タ，すなわち以下の $R_1(t_1)$ で表現される．

$$R_1(t_1) = \{r | r \in R \wedge r[T_a] \leq t_1 \wedge t_1 < r[T_d]\} \tag{3.2}$$

図 3.1 の (a) にトランザクション時間として 7 月 1 日を指定したスナップショットを，(b) に 7 月 2 日を指定したスナップショットを示す．バイテンポラルデータベースでは各データは，トランザクション時間と有効時間という，直交する 2 つの時間軸によって表現されるため，トランザクション時間のみを指定したスナップショットでは，有効時間に関する履歴が検索される．ここで，各々のスナップショットでは指定されたトランザクション時間のデータベースの状態が示される．例えば，図 3.1(a) では，有効時間が 7 月 1 日のデータ $ID = 001, 002, 003, 004$ が検索され，7 月 2 日に行われた訂正は反映されない．一方，図 3.1(b) ではデータ $ID = 002, 003, 004$ 対して 7 月 2 日に行われた変更，削除，追加が反映されており，また，7 月 2 日に入力されたデータ $ID = 005$ も検索される．

次に，有効時間 t_2 のみを指定したスナップショットを以下に示す．このスナップショットは，有効時間 t_2 を含む時間区間 $V = \{V_a, V_d\}$ となるが，$V_a = V_d$ であるため，以下の $R_2(t_2)$ で表現される．

$$R_2(t_2) = \{r | r \in R \wedge r[V_a] = t_2\} \tag{3.3}$$

図 3.2 の (a) に有効時間として 7 月 1 日を指定したスナップショットを，(b) に 7 月 2 日を指定したスナップショットを示す．これは，有効時間，すなわち実世界での入金日を指定

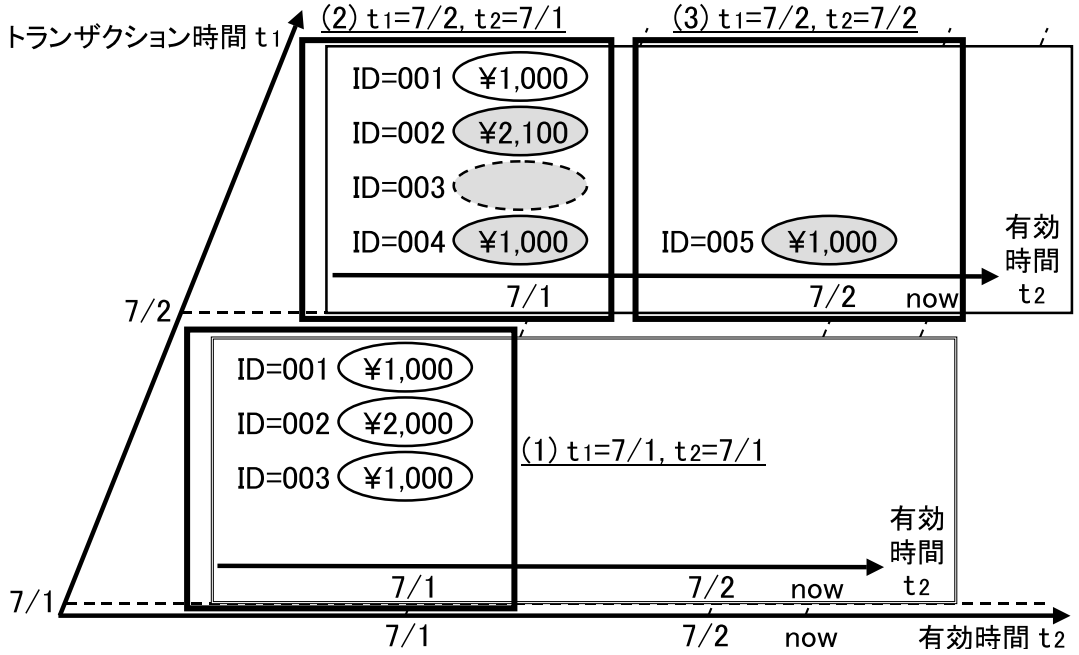


図 3.3: トランザクション時間と有効時間を指定した入金テーブルのスナップショット

した検索になる．(a) では 7 月 1 日入金のデータの，トランザクション時刻に伴う変更の履歴が検索される．ここで，(a) の $ID = 005$ のデータは 7 月 1 日の入金データとは区分され，別の有効時間，すなわち 7 月 2 日の入金データとして検索される．

最後に，トランザクション時間 t_1 と有効時間 t_2 の両方を指定したスナップショットを以下に示す．スナップショットは，トランザクション時間 t_1 を含む時間区間 $T = \{T_a, T_d\}$ と有効時間 t_2 を含む時間区間 $V = \{V_a, V_d\}$ を持つデータで示される．ここで，式 (3.3) と同様に $V_a = V_d$ のため，このデータは以下の $R_3(t_1, t_2)$ で表現される．

$$R_3(t_1, t_2) = \{r | r \in R \wedge r[T_a] \leq t_1 \wedge t_1 < r[T_d] \wedge r[V_a] = t_2\} \tag{3.4}$$

このスナップショットは，2 つの時間軸の両方を指定しているため，履歴ではなく，指定時間の状態が検索される．図 3.1 において，図 3.3 の (1)，(2)，(3) にトランザクション時間と有効時間として，各々，7 月 1 日あるいは 7 月 2 日を指定した検索結果を示す．なお，トランザクション時間が 7 月 1 日で，有効時間が 7 月 2 日のデータは該当データがないため検索されない．図に示されるように，各データの持つトランザクション時間区間 T と，有効時間の時間区間 V，すなわちシステムに入力あるいは訂正された日と，入金された日毎にデータを分けて検索することが可能になる．ここで，(1) では 7 月 1 日の入金直後のデータ，(2) では 7 月 1 日の入金に対して 7 月 2 日に行われた変更，削除，追加を反

映したデータ，(3) では 7 月 2 日の入金データが検索される．

なお，有効時間が時間区間であるデータ，すなわち $V_a < V_d$ であるデータを扱う場合には，式 (3.3) の $R_2(t_2)$，式 (3.4) の $R_3(t_1, t_2)$ は各々，以下で表現される．

$$R_2(t_2) = \{r | r \in R \wedge r[V_a] \leq t_2 \wedge t_2 < r[V_d]\} \tag{3.5}$$

$$R_3(t_1, t_2) = \{r | r \in R \wedge r[T_a] \leq t_1 \wedge t_1 < r[T_d] \wedge r[V_a] \leq t_2 \wedge t_2 < r[V_d]\} \tag{3.6}$$

3.2.3 バッチ検索処理における効果

バイテンポラルデータベースにおいて，トランザクション時間と有効時間の両方を指定したスナップショット，すなわち式 (3.4) による検索が，2.2 節に示した以下のバッチ検索処理の要件のうち，オンライン入力との並行実行（要件 1），データ訂正時の再実行（要件 2），スケジュールの柔軟性（要件 3）を満たすことを示し，併せて，2.4.2 節に示したトランザクション時間データベースの課題を解決できることを示す．

■ 要件 1：オンライン入力との並行実行

バイテンポラルデータベースはトランザクション時間データベースの機能を含んでいる．すなわち，3.2.2 節の式 (3.4) に示すトランザクション時間を指定したスナップショットでは過去のトランザクション時間のデータを検索する．従って，2.4.1 節で示したトランザクション時間データベースのスナップショットの検索と同様に，オンライン入力中でも，その影響を受けない一貫した検索結果を得ることができる．

■ 要件 2：データ訂正時の再実行

式 (3.4) および図 3.3 に示すように，バイテンポラルデータベースの検索では，データがデータベースに存在していた時刻であるトランザクション時間 t_1 と，実世界で有効だった時刻である有効時間 t_2 の両方を指定したスナップショットを検索できる．従って，有効時間 t_2 により検索対象データを指定し，時刻 t にデータを訂正した場合には，$t_1 > t$ なるトランザクション時間 t_1 を指定して検索を行うことにより，訂正後のデータの検索が可能になる．

図 3.4 に，図 2.13 においてトランザクション時間 t_1 と有効時間 t_2 を指定して検索した結

データベースの状態(7/2現在)

現金総額テーブル

入金日	Ta	Td	Va	Vd	金額
7/1	7/1	now	7/1	7/1	6,600
7/2	7/2	now	7/2	7/2	1,000

入金テーブル

ID	Ta	Td	Va	Vd	金額	(1)	(2)	(3)	区分
001	7/1	now	7/1	7/1	1,000	●	●		
002	7/1	7/2	7/1	7/1	2,000	●			変更前
002	7/2	now	7/1	7/1	2,100		●		変更後
003	7/1	7/2	7/1	7/1	1,000	●			削除
004	7/2	now	7/1	7/1	3,500		●		追加
005	7/2	now	7/2	7/2	1,000			●	7/2の入力

入金テーブルのスナップショット

(1) t_1=7/1, t_2=7/1

ID	Ta	Td	Va	Vd	金額
001	7/1	now	7/1	7/1	1,000
002	7/1	7/2	7/1	7/1	2,000
003	7/1	7/2	7/1	7/1	1,000
				合計	4,000

(2) t_1=7/2, t_2=7/1

ID	Ta	Td	Va	Vd	金額
001	7/1	now	7/1	7/1	1,000
002	7/2	now	7/1	7/1	2,100
004	7/2	now	7/1	7/1	3,500
				合計	6,600

(3) t_1=7/2, t_2=7/2

ID	Ta	Td	Va	Vd	金額
005	7/2	now	7/2	7/2	1,000
				合計	1,000

図 3.4: バイテンポラルデータベースによるデータ訂正時の再実行

果を示す．これは，図 3.3 に示した検索の概念について，各データの属性値を明示したものであり，(1) は t_1 と t_2 が共に 7 月 1 日 の検索結果を，(2) は $t_1 =$ 7 月 2 日 で $t_2 =$ 7 月 1 日 の検索結果を，(3) は t_1 と t_2 が共に 7 月 2 日 の検索結果を示す．また,「●」は各検索結果に該当するデータを示している．この例に示すように，有効時間により入金日を，トランザクション時間により入力あるいは訂正した日を指定できる．従って，図 3.4 の (2) に示す，$t_1 =$ 7 月 2 日，$t_2 =$ 7 月 1 日 を指定して式 (3.4) の検索を行うことにより，7 月 2 日 に実施した変更，削除，追加の訂正を反映すると共に，7 月 2 日の入金データ $ID = 005$ を反映しない状態，すなわち訂正後の 7 月 1 日現在の状態を検索することが可能になる．

このように，オンライン入力中であっても，指定した有効時間の入力データに対して，訂正を反映した上で再検索を行うことが可能であり，要件 2 を満足したバッチ検索処理が可能になる．

■ 要件 3：スケジュールの柔軟性

バイテンポラルデータベースはトランザクション時間を管理しているため，過去のトランザクション時間を指定した検索が可能になる．すなわち，特定時刻のデータを利用する処理であっても，特定時刻以降の任意の時刻に該当処理を実行することにより負荷の分散を図ることができる．

ここで，トランザクション時間データベースでは，図 2.14 に示すように過去の指定時刻の検索を行う場合には，それ以降の訂正が反映されないため，現在の直近の時刻での検索結果との差異が，指定時刻以降の入力の状況と整合しないという課題があった．これに対し，バイテンポラルデータベースでは要件 2 で示したように，対象データの有効時間と，現在の直近の時刻をトランザクション時間として指定することにより，現在までに実行された訂正を反映した検索を行うことができる．すなわち，処理を実行するタイミングを遅らせたり，複数の時刻に実施すべき処理を同時に実行したりしても，データの訂正を反映した一貫性のある結果を得ることができる．

図 3.5 に，図 2.14 に示したトランザクション時間データベースによる事例を，バイテンポラルデータベースにより実施した場合を示す．図 3.5 で，(1)，(2)，(3) は図の「入金テーブルのスナップショット」に示す条件での検索結果を,「入金テーブル」の「●」は図 3.4 と同様に検索結果に該当するデータを示す．バイテンポラルデータベースでは，有効時間として入金日が表現される．従って，例えば訂正によって追加されたデータ $ID = 204$ の入金日は 7 月 30 日であり，トランザクション時間 T_a が 8 月 20 日であるにもかかわら

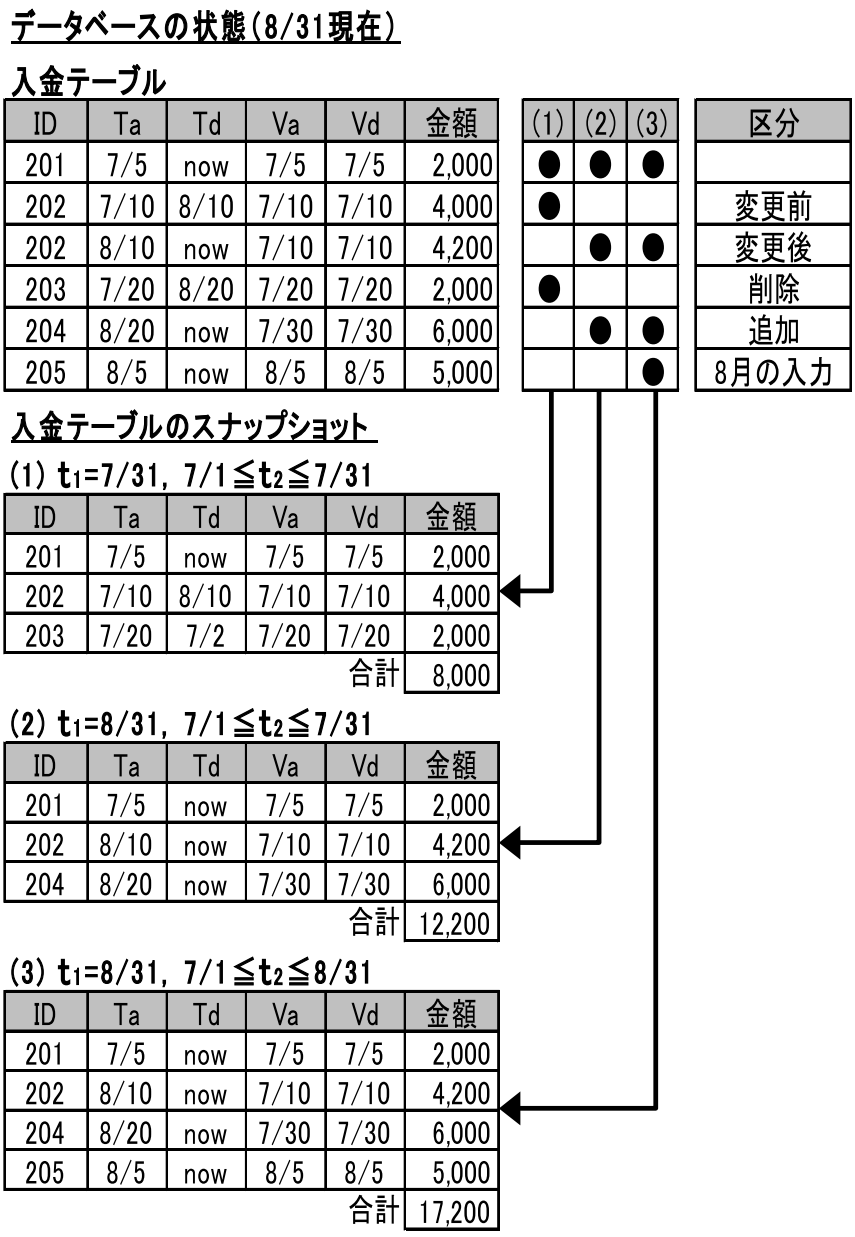


図 3.5: バイテンポラルデータベースによるスケジュールの柔軟性

ず7月の入金データであること，すなわち，8月入金のデータである $ID = 205$ とは入金日の属する月が別であることが識別できる．

図 3.5 の (1) にトランザクション時間を 7 月 31 日としたスナップショットにおける 7 月の入金状況を，(2) に同じく 8 月 31 日とした場合の入金状況を示す．この検索結果は，式 (3.4) において $r[V_a] = r[V_d]$ であることから， $r[V_a]$ が指定された有効時間区間のデータとなる．すなわち，スナップショットのリレーションは以下で表現される．

$$R_4(t_1, t_{2b}, t_{2e}) = \{r | r \in R \wedge r[T_a] \leq t_1 \wedge t_1 < r[T_d] \wedge t_{2b} \leq r[V_a] \wedge r[V_a] < t_{2e}\} \qquad (3.7)$$

ここで，$[t_{2b}, t_{2e})$ は有効時間の範囲を示し，t_1 はトランザクション時間を示す．

図 3.5(2) のスナップショットは，有効時間の範囲が $[7/1, 7/31)$ であり，トランザクション時間を 8 月 31 日として検索するため，リレーションは $R_4(8/31, 7/1, 7/31)$ となる．従って，8 月 31 日までに実施された変更，削除，追加の訂正が反映される．また，図 3.5(3) にスナップショットは，有効時間の範囲が $[7/1, 8/31)$ であり，トランザクション時間を同じく 8 月 31 日にした，7 月と 8 月の入金の状況を示す．図 3.5(2) のトランザクション時間を 8 月 31 日にした検索では，8 月 31 日までの全ての訂正が反映されている．従って，(2) と (3) の差異は，8 月の入金データである $ID = 205$ のデータとなり，8 月 31 日と 7 月 31 日の入金総額の差異は 8 月の入金となって，一貫性のある結果を得ることができる．

このように，要件 3 を満足した柔軟なバッチ処理のスケジューリングが行えるだけでなく，データの誤りを訂正した場合であっても訂正結果を反映した一貫性のある検索結果を得ることができる．

3.3 基幹系システムへの適用

本節では，実際の業務システムへの適合性（要件 4）を評価するため，バイテンポラルデータベースを基幹系システムである自治体システムに適用した結果を示す [106]．ここで，実装にあたっては，バイテンポラルデータベースの実装上の課題であるデータ量の増大と検索処理の複雑化への対策が必要だった．本適用システムは約 40 の自治体に展開，運用されている．ここでは実際の自治体システムの構築，およびシステム運用結果に基づき，実装の方式と業務システムへの適合性について示す．

3.3.1 自治体システムの概要

自治体システム [106, 107] は自治体の行政事務 [75] を支援するシステムであり，図 3.6 に示す様に各種の業務システムから構成される．大きくは以下のように区分される．

(1) **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，およびこれに関する資格管理，統計資料の作成を支援する．また，台帳に基づく該当自治体での選挙権の登録，管理や，学校教育，成人式に関する台帳管理，通知業務を支援する．

(2) **税関連システム**　自治体管掌の地方税に関する課税，税金の徴収事務，税証明の発行業務，およびこれに関する各種の統計資料の作成業務を支援する．

(3) **福祉系システム**　保育所，児童手当などの福祉行政に関する資格管理，料金の徴収事務，給付事務，およびこれに関する各種の統計資料の作成業務を支援する．

(4) **内部情報系システム**　人事管理，給与支給，財務会計事務，起債管理などの，自治体内部における事務業務，およびこれらに関する各種の統計資料の作成業務を支援する．

自治体システムでは，住民からの届出や申請，あるいは庁内での事務遂行に際し，各種のデータがオンライン入力される．オンライン入力は，本庁だけでなく，支所，あるいは公共施設などの出張所に設置された端末から行われ，住民からの請求や，事務上必要がある場合には入力データを反映した証明書や伝票などが即時に発行されるため業務時間中は中断できない．また，統計資料などの大量のデータを検索する必要のある処理は，バッチ処理で行われる．

各業務システムにおいてオンライン入力されるデータ，即時発行およびバッチ処理で作成される業務用帳票を表 3.1 に示す．また，図 3.7 に住民記録システム [66] における運用の流れを示す．自治体の窓口では，住民からの異動届が受付けられ，異動データがオンライン入力されてデータベースに蓄積される．このとき，必要な証明書，例えば転出の際の転出証明書や，交付請求のあった住民票は届出の内容を反映して即時に発行される．一方，バッチ処理によって作成される人口統計表や，人口動向調査報告書は，指定された期日の業務終了時点現在のデータで作成する必要がある．また，資料の作成にあたっては，データの確認が行われ，データの誤りがあった場合には訂正が行われる．

図 3.6: 自治体システムの構成

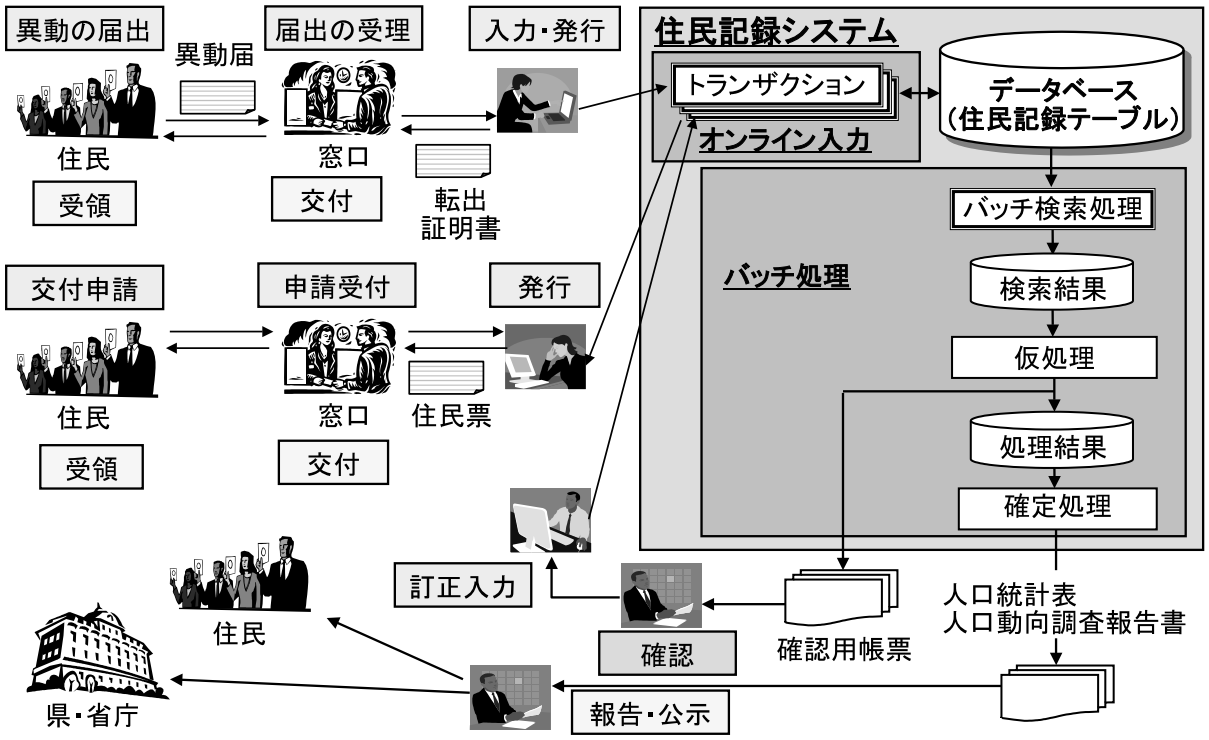


図 3.7: 住民記録システムの運用

このように，自治体システムの各業務においてもオンライン入力とバッチ処理の両方の処理形態があり，システム運用の効率化のためには両者を並行して実行できることが必要になる．一方，両者を並行して実行するためには，2.2 節に示したバッチ検索処理の要件 1，2，3 を満たす必要があった．

まず，オンライン入力との並行実行（要件 1）については，例えば統計資料の作成で大量のデータを検索するため処理に時間を要する．一方，オンライン入力は頻繁に行われるため，検索中にデータが更新される．従って，オンライン入力中にもバッチ検索処理で一貫性のある検索が行えることが必要になる．

次に，データ訂正時の再実行（要件 2）については，オンライン入力に際してはデータベース，システムの入力機能によるデータチェック，および職員の目視による確認により，データの一貫性はある程度確保される．しかし，テーブル相互の整合確認や，集計資料と個々のデータ集計などの突合せによる確認は，効率の点からバッチ処理で行う必要がある．このため，バッチ処理では必要に応じて事前にチェックリストの作成や，仮処理を行ってデータの確認が実施され，データの誤りを検出した場合には，データ訂正を行った上で，再度，バッチ検索処理を実行する必要があった．

表 3.1: 各業務システムにおける入力データと業務用帳票の例

No	**業務区分**	**区分**	**入力データと業務用帳票の例**
(1)	住民情報系システム	入力データ	住民の異動（出生，転入，転居，転出など）
		即時発行帳票	住民票，転出証明書
		バッチ処理帳票	人口統計表，人口動向調査報告書
(2)	税関連システム	入力データ	課税物件の異動（軽自動車の登録など）
		即時発行帳票	標識交付証明書，納税証明書
		バッチ処理帳票	登録台数調べ，課税状況調べ
(3)	福祉系システム	入力データ	老人医療保険資格の取得，喪失
		即時発行帳票	老人医療受給者証
		バッチ処理帳票	異動状況集計表，負担区分集計表
(4)	内部情報系システム	入力データ	財務会計の執行データ
		即時発行帳票	財務会計伝票
		バッチ処理帳票	日計表，月計表，決算資料，決算統計資料

なお，データの訂正は，証明書などの発行状況に応じて，内部処理での訂正と，業務での訂正の，２つの方法が取られた．内部処理での訂正は，入力データの利用前に行う訂正であり，例えば，住民の異動の入力においては，証明書発行前に確認を行い，誤りが検出されれば訂正する．この場合には，住民への通知などの業務としての訂正手続きや，住民票への履歴の記載は行われない．一方で，住民票などの証明書発行後に入力誤りが検出された場合には，職権修正などの業務での訂正手続きが行われ，住民票に履歴が記載されて住民へ通知された．

スケジュールの柔軟性（要件 3）については，統計資料や集計表は指定された期日に定期的に作成され，都道府県や関係機関に報告された．例えば，人口統計表は毎月初日の業務終了時点のデータで作成され，決算関係の資料は毎月末日の業務終了時点のデータで作成された．指定期日は，月末，月の初日に集中しているが，報告は必ずしも翌日の朝に行うわけではない．従って，業務負荷の平準化の点から，報告までのスケジュールに応じて優先順位を付けて，順次，処理を行うことのできる方式が必要になった．なお，従来のオンライン入力とバッチ処理の時間帯を分けた運用では，これらの期日にはオンライン入力終了を待ってさまざまなバッチ処理が行われたため，夜間バッチの時間が長くなるという

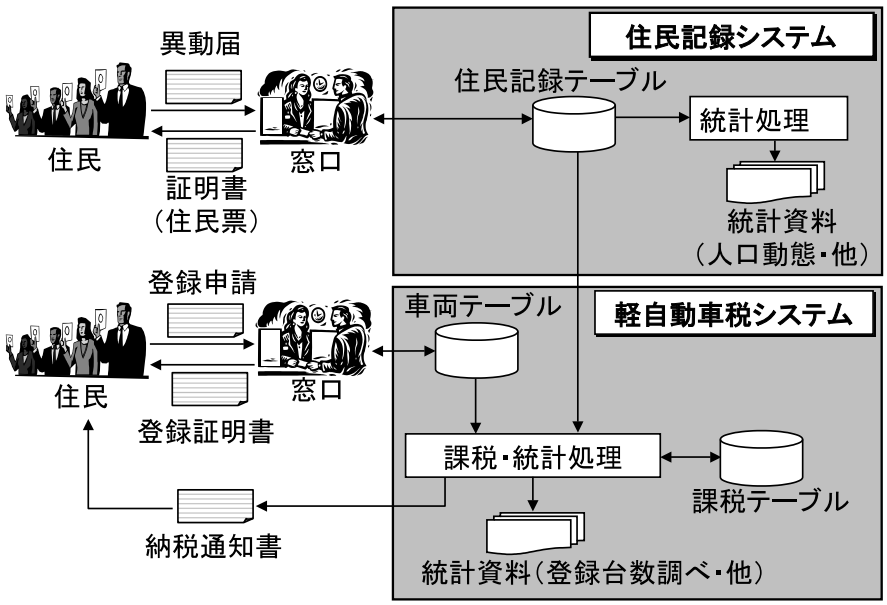


図 3.8: 自治体システムのデータフロー

問題があった.

また，各業務システムでは時系列のデータ履歴管理が必要になった．以下に管理する履歴データの例を示す.

- **住民の異動履歴**　出生，転入から，転居，婚姻などを経て，死亡，転出に至るまでの住民の異動履歴.
- **課税に関する履歴**　軽自動車などの課税対象物件の登録，譲渡，廃車などの異動履歴，課税および徴収の履歴.
- **福利の資格・徴収・給付履歴**　福祉行政に関する資格の取得，喪失の履歴，保険料や料金の調定，徴収の履歴，給付の履歴.
- **職員に関する履歴**　自治体職員の人事異動履歴，勤怠・賞罰の履歴，給与支給の履歴.

業務におけるデータフローの例として，図 3.8 に住民記録と軽自動車税 [67] の例を示す. 軽自動車の登録，廃車の申告は，住民の異動の届出と同様に窓口で受付けられ，オンライン入力されてデータベースに蓄積された．統計処理，課税処理などの大量のデータ検索を伴う処理はバッチ処理で処理され，図 3.8 の住民記録テーブルに示されるように，蓄積されたデータは，各業務で相互に活用された．従って，データ誤りが検出された場合には,

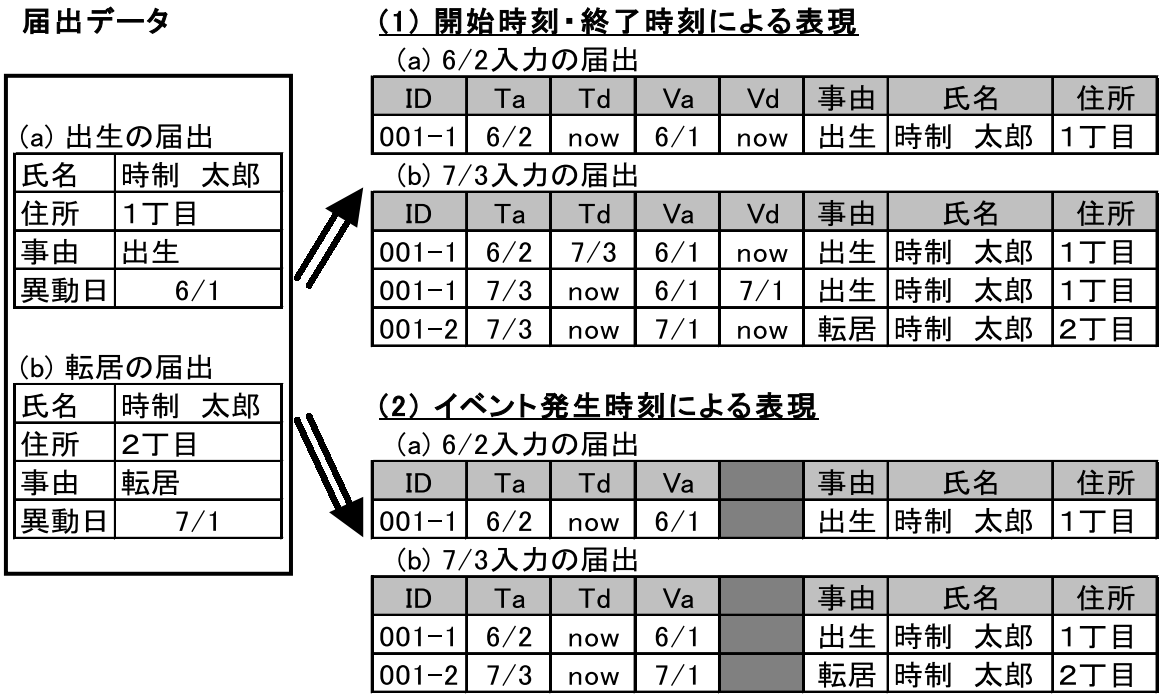


図 3.9: 有効時間属性の表現

該当バッチ検索処理の検索結果だけを直接訂正するなどの暫定的な手段による訂正では他の処理と結果が整合しなくなる．従って，データの誤りはデータベースのテーブルを，オンライン入力によって訂正することが必要になった．

3.3.2 バイテンポラルデータベースの実装

商用のリレーショナルデータベース [27] を使用し，各テーブルに必要に応じてトランザクション時間，有効時間の属性を付加してバイテンポラルデータベースを構成した．バイテンポラルデータベースの実装においては，履歴管理に伴うデータ量の増大という課題があるため，これへの対応が必要になった．

このため，適用システムでは有効時間属性の実装にあたっては，下記の「イベント発生時刻による表現」を基本にした．なお，従来のバイテンポラルデータベースでは，下記の「開始時刻・終了時刻による表現」が基本になる．

- **属性の表現**
  - **イベント発生時刻による表現**

住民票の様に，出生，転居などのイベント発生により対象の状態が変化し，次のイベント発生まで前回の状態が保持されるものは，イベント発生時刻で有効時間属性を表現した．

- **開始時刻・終了時刻による表現**
  口座振替期間のように予め開始，終了期日を登録，管理する必要があるか，あるいは資格の取得・喪失の様に開始，終了をセットで管理する必要のあるテーブルは，開始時刻・終了時刻で有効時間属性を表現した．

開始時刻・終了時刻による表現では，図 3.9 の (1) に示す様にデータの変更を行う場合には，以下の操作が行われ，２レコードが追加される．図で，T_a，T_d，V_a，V_d は 3.2.2 節の表記と同じである．

- $ID = 001 - 1(T_a = 6/2)$：変更元データに削除時刻 T_d を設定
- $ID = 001 - 1(T_a = 7/3)$：変更に対応した履歴データとして終了時刻 V_d を設定したデータを追加
- $ID = 001 - 2$：住所が２丁目に変更されたデータを追加

ここで，$ID = 001 - 1$ の $T_a = 6/2$ のデータはトランザクション時間が 6 月 2 日から 7 月 2 日までの状態を示す履歴データであり，$ID = 001 - 1$ の $T_a = 7/3$ は 7 月 3 日以降に業務で使用される履歴データになる．

一方，イベント発生時刻による表現では，変更の際には図 3.9 の (2) に示す様に変更後のデータのみを追加することでデータの増加を抑制した．この場合，指定した時刻の実世界での状態は，指定時刻の直前の開始時刻を持つデータにより表現される．従って，トランザクション時間 t_1，有効時間 t_2 を指定したスナップショットは以下で表現される．

$$\begin{aligned} R_5(t_1, t_2) = &\{r | r \in R \wedge r[T_a] \leq t_1 \wedge t_1 < r[T_d] \\ &\wedge r[V_a] = max\{s[Va] | s \in R \wedge s[V_a] \leq t_2\}\} \end{aligned} \tag{3.8}$$

このように，イベント発生時刻による表現を基本とし，開始時刻と終了時刻の両方を管理する必要のあるテーブルについてのみ開始時刻・終了時刻による表現を適用することで，履歴データの増大を抑止した．

また，時間属性の粒度については，トランザクション時間はデータベース上のキー属性であるために，データ更新の頻度から設定する必要があった．本システムでは，画面からのデータ入力に少なくとも数秒を要するため，時間の単位はこれ以下にする必要があっ

た．このため，トランザクション時間の単位は１秒にした．有効時間は業務に関連するため，業務で必要な単位で管理した．以下に有効時間の単位の例を示す．

- **有効時間の単位**
  - **時・分**　職員の勤怠
  - **日**　住民の異動，課税対象物件の異動，福利資格の取得・喪失，口座振替期間，職員の異動
  - **月**　職員の給与支給情報
  - **年・年度**　軽自動車税，その他の税額，および税額算定のための収入・所得額

さらに，実際の運用では，実世界の状態がシステムに反映されるのに時間を要するという課題があった．このため，当日の業務終了後の統計を翌日報告する業務のうち，実世界の状態がシステムに即時に反映されないデータを扱うものは，自治体への届出日を基準にして統計を作成した．例えば，住民の出生，転居，転入の異動の届出は 14 日以内あり，実世界の状態がシステムに反映されるまで時間を要する．しかし，人口動態などの統計は指定期日の業務終了後の状態を，すみやかに報告する必要がある．このため，住民記録テーブルでは有効時間である異動日と併せて届出日を管理した．この届出日は，時制データベースの時間軸のうち，利用者によって与えられる時間軸であるユーザ定義時間に該当する．

図 3.10 にユーザ定義時間である，届出日によるスナップショットの例を示す．図で，N_a，N_d が届出日の開始時刻と終了時刻を示す．ここで，終了時刻は次の届出の届出日になる．すなわち，届出日の時間区間属性は有効時間と同様であり，有効時間と同様の検索ができる．図では届出日が 6 月 2 日であるデータに対し，6 月 3 日に訂正を行った例を示す．図の (1) では，届出日に 6 月 2 日，トランザクション時間に 6 月 2 日を指定することで訂正前の検索結果が，(2) では届出日に同じく 6 月 2 日を，トランザクション時間に 6 月 3 日を指定することで，$ID = 002 - 1$ の変更，$ID = 003 - 1$ の削除，$ID = 004 - 1$ の追加という全ての訂正が反映された検索結果が得られた．

3.3.3　バッチ処理の実装

バッチ処理の実装については，バイテンポラルデータベースの履歴管理による検索の複雑化という課題があるため，これへの対応が必要になった．このため，適用システムでは

データベースの状態

6/2の状態　　[Na, Nd):ユーザ定義時間(届出日)

ID	Ta	Td	Va	Vd	Na	Nd	事由	氏名	住所
001-1	6/2	now	6/1	now	6/2	now	出生	時制　太郎	1丁目
002-1	6/2	now	5/31	now	6/2	now	転入	変更　次郎	2丁目
003-1	6/2	now	5/21	now	6/2	now	出生	削除　三郎	3丁目

6/3の状態

ID	Ta	Td	Va	Vd	Na	Nd	事由	氏名	住所	(1)	(2)
001-1	6/2	now	6/1	now	6/2	now	出生	時制　太郎	1丁目	●	●
002-1	6/2	6/3	5/31	now	6/2	now	転入	変更　次郎	2丁目	●	
002-1	6/3	now	5/31	now	6/2	now	転入	変更　次郎	5丁目		●
003-1	6/2	6/3	5/21	now	6/2	now	出生	削除　三郎	3丁目	●	
004-1	6/3	now	5/21	now	6/2	now	転入	追加　四郎	4丁目		●

スナップショット

(1) トランザクション時間=6/2, 届出日=6/2

ID	Ta	Td	Va	Vd	Na	Nd	事由	氏名	住所
001-1	6/2	now	6/1	now	6/2	now	出生	時制　太郎	1丁目
002-1	6/2	6/3	5/31	now	6/2	now	転入	変更　次郎	2丁目
003-1	6/2	6/3	5/21	now	6/2	now	出生	削除　三郎	3丁目

(2) トランザクション時間=6/3, 届出日=6/2

ID	Ta	Td	Va	Vd	Na	Nd	事由	氏名	住所
001-1	6/2	now	6/1	now	6/2	now	出生	時制　太郎	1丁目
002-1	6/3	now	5/31	now	6/2	now	転入	変更　次郎	5丁目
004-1	6/3	now	5/21	now	6/2	now	転入	追加　四郎	4丁目

図 3.10: ユーザ定義時間（届出日）によるスナップショット

中間ファイルをデータベースのテーブルで構成し，データベースから抽出したデータについても，引き続きデータベースの検索機能を利用して処理することで，個々の検索機能の複雑化を抑止した．以下に具体的な実装方式を示す．

適用システムのバッチ処理は，図 3.7 に示すようにバッチ検索処理，および仮処理や確定処理における帳票作成や外部に渡すデータ作成の機能によって構成した．また，図 1.2 に示したように，バッチ処理は長時間に及ぶため，ジョブネットを構成して実行状況の監視や実行制御を行う必要がある．適用システムでは，バッチ処理を以下の実行基盤の上に構築した．

- **バッチ処理の運用管理**
  バッチ処理の監視，実行制御は商用のバッチ処理管理ミドルウェア [104] を使用し，ジョブネットとしての実行環境を構成した．ミドルウェアの機能により，プログラムあるいは SQL 文を，ジョブに組み込んで順次，実行すると共に，実行時のパラメータを外部から渡す構成にした．

- **中間ファイル**
  中間ファイルは通常，汎用機やオフコンでは順次編成（SAM）ファイルで構築される．適用システムでは，中間ファイルもデータベースのテーブル（以下，ワークテーブルと略記）で構築した．これは，バッチ処理の各プログラムをデータベースの検索機能を使用して構築することにより，段階的にデータを処理して各プログラムでの検索機能を単純化することを狙ったことによる．

- **プログラム**
  プログラムは一部の機能を除き，SQL あるいは SQL/PSM（Structured Query Language/Persistent Stored Modules）[94] で作成した．これらに対してバッチ処理管理ミドルウェアを通じてパラメータを渡したが，SQL 文や SQL/PSM の構文の一部をパラメータ化することで，プログラムそのものを変更し，必要に応じて柔軟な検索を実行できるようにした．

- **帳票印刷**
  帳票に関しては，帳票出力管理ミドルウェア [105] を使用し，これによって印刷状況の監視や，プリンタに関する障害発生時の再印刷を可能にした．また，このミドルウェアによりマスタのテーブルおよびワークテーブルから直接帳票を印刷できた．

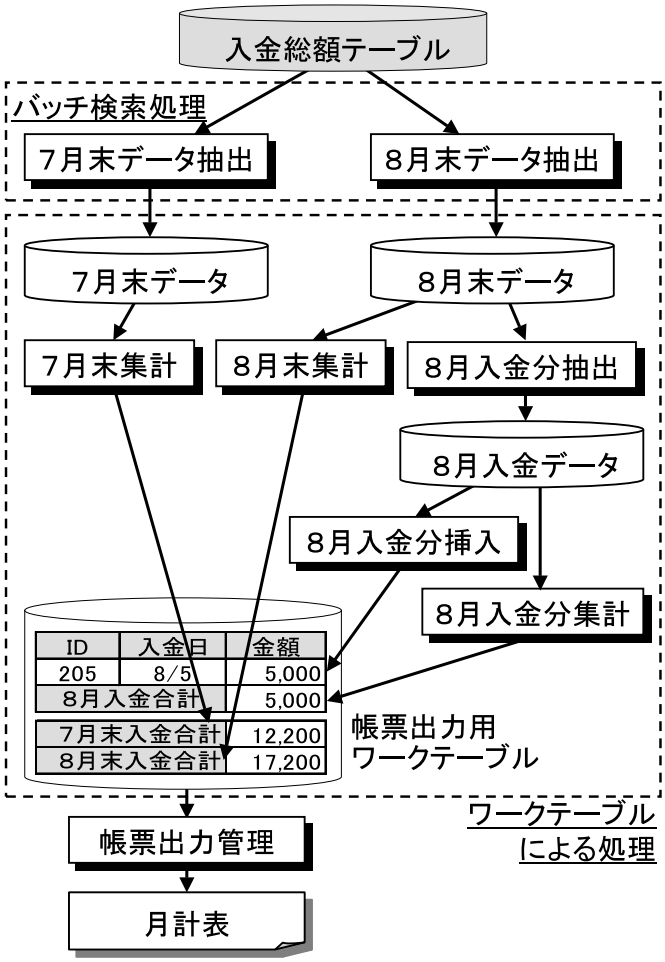


図 3.11: バッチ処理の構成

図 3.11 にバッチ処理の構成を示す．中間ファイルをワークテーブルで構成したことにより，バッチ検索処理以降のプログラムでも引き続きデータベースの機能である SQL や SQL/PSM を使用したデータ処理ができた．このため，バッチ検索処理のプログラムを始めとして個々のプログラムを簡素化し，段階的にデータを処理して最終的な結果を得るという構成が可能になった．例えば，バイテンポラルデータベースでは，図 2.8 に示すように履歴管理に伴いテーブルの結合演算が複雑化する．しかし，段階的に処理することにより，例えば，バッチ検索処理ではマスタとなるテーブル毎に必要なスナップショットのみを検索し，結合演算などのデータ処理は以降のプログラムで実行するという構成が可能になった．

さらに，バッチ処理では，プログラムを標準化して共通的に使用するため，極力，単機能のプログラムによって検索処理を行い，その入出力対象となるワークテーブルのリレーションも共通化した．さらに，帳票印刷についてもデータの処理は極力プログラムで実行し，帳票印刷前に帳票に準拠したワークテーブルとして作成する方式にした．こうして，帳票印刷のミドルウェアに依存するデータ処理を極力少なくした．

この構成のため，プログラムは入力用のワークテーブルからデータを検索して次のワークテーブルへ出力するという，SQL 文の insert を多用する構成になった．データ量が多い場合や，複雑な検索の対象となるワークテーブルでは検索の高速化のためインデックスが必要になるが，反面，インデックスを使用すると insert は遅くなる．特に，バイテンポラルデータベースでは履歴を扱うため，スナップショットデータベースの場合よりもデータ量が多くなる．このため，データ量の多い処理はインデックスなしで insert，あるいはデータをインポートして，その後にインデックスを生成する方式にした．

なお，ワークテーブルの作成，削除の運用は，従来の汎用機やオフコンの中間ファイルの運用に準じ，ジョブの先頭で使用するワークファルを全て削除した上で再作成し，ジョブの最後で削除する運用にした．ここで，先頭で削除するのは従来の運用と同様，前回，異常終了した場合に備えるものである．

3.3.4 オンライン入力の実装

窓口で受付けた届出の内容は業務画面からオンライン入力した．この入力は業務的に以下の特徴があった．

- 住民の届出により入力を行うため，同一住民に対する複数端末からの同時入力は通

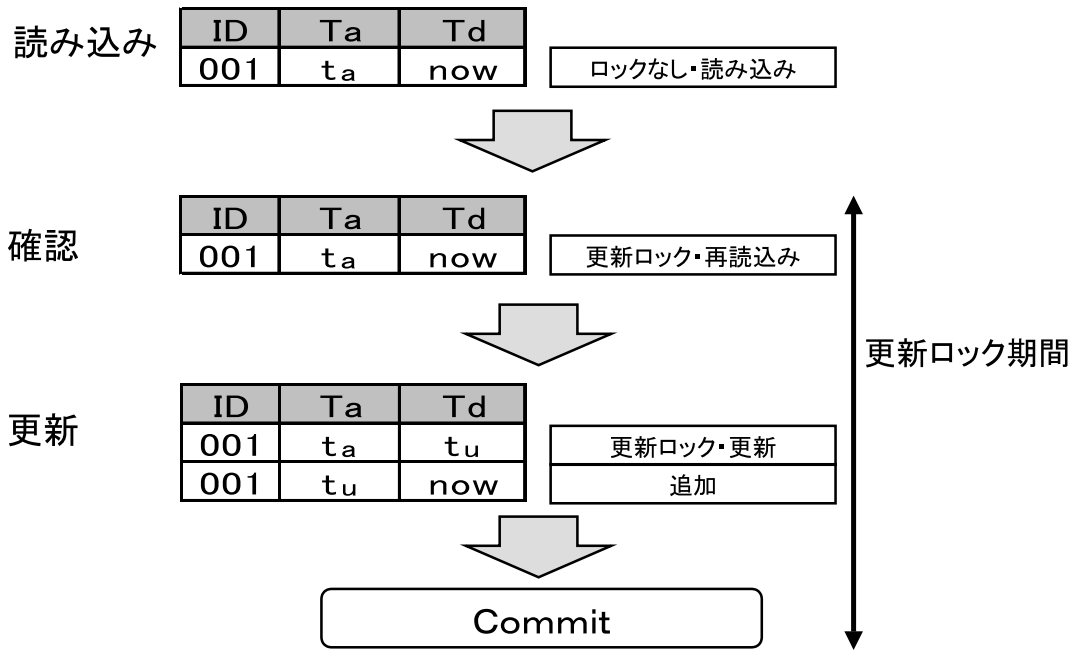


図 3.12: バイテンポラルデータベースにおける楽観的方法による更新

常業務では起こりえない

- 窓口では操作の際に届出内容の確認を行うため入力時間が長くなる

このため，2.3 節に示した楽観的方法を応用して，図 3.12 の手順でトランザクション時間の開始時刻，終了時刻を利用した更新を行い，ロック時間を短縮した．以下にその手順を示す．図 3.12 に示されるように，リソースに対する更新ロックは更新前のデータに対して行われる．

- **読み込みフェーズ**
  - ロックせずにテーブルから更新前データを読み出し．
  - 読み出したデータを画面で修正し，更新後データを作成．
- **確認フェーズ**
  - 同一の検索条件で，データベースのレコード更新ロックにより更新前のデータを再読み出し．
  - 該当データが既にロックされている場合，追加時刻が更新されている場合は，他のトランザクションが更新中あるいは更新済のためロールバックを行う．そ

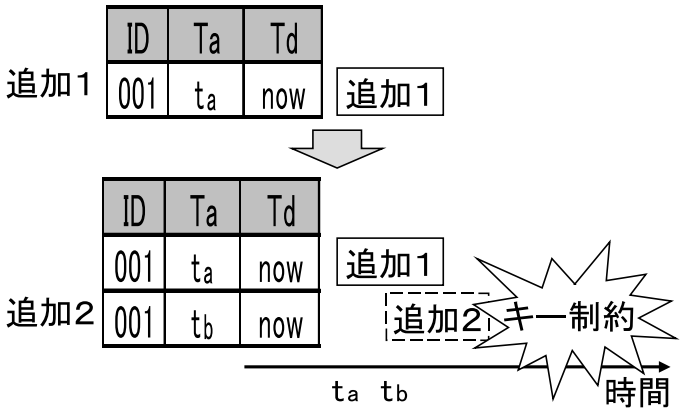


図 3.13: 楽観的方法におけるキー制約による重複登録の防止

れ以外の場合には，更新フェーズを実行する．

- **更新フェーズ**
  - 再読み出しした更新前データに削除時刻を設定し，データベースを更新．
  - データベースに更新後データを追加．このとき，追加時刻には更新前データの削除時刻を，削除時刻には now を設定する．
  - エラーが発生した場合にはロールバックを，それ以外の場合にはコミットを行いトランザクションを完了する．

ここで，２箇所以上の端末からデータベースに対し同一のデータを新たに追加する場合には，更新前データがないためロックがかからない．また，追加時刻が異なると時制データベースでは別データとして扱われる．このため，トランザクション時間については追加時刻ではなく，削除時刻をデータベースの主キー属性とし，同一データの二重登録はキー制約によって防止する構成にした．図 3.13 に統一のデータを二重登録しようとした場合のデータベースの状況を示す．ID と削除時刻 T_d を主キーとすることによって，同一 ID のデータのである，追加１と，追加２のデータの二重登録は，キー制約によって防止される．

3.3.5 データ訂正の運用

3.3.1 節に示すように，データの訂正は以下の２種類で運用された．

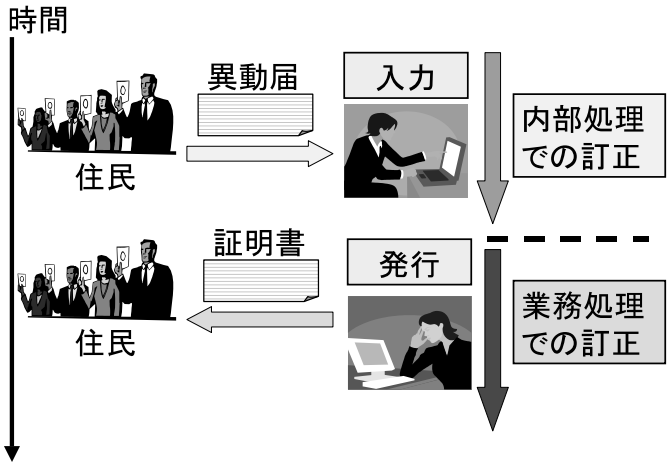


図 3.14: 住民票に関する訂正の運用

- **内部処理での訂正**　誤入力に対し内部処理の段階で行うデータの訂正
- **業務処理での訂正**　通常の業務手続として行うデータの訂正

住民票に関する内部処理での訂正，および業務処理での訂正の運用を，図 3.14 に示す．異動届によって受付けられたデータを入力した場合，誤った入力を行うことがある．誤りが検出された段階が住民票などの証明書発行前であれば，内部処理での訂正として住民票の履歴表示を行わない訂正が実行された．一方，証明書の発行後であれば，業務処理での訂正として，証明書への履歴の記載などの業務的な手続きが必要になった．これらの訂正は，バイテンポラルデータベースの時間属性を利用して，別個の訂正として実装した．このため，業務画面についても，両者は別の入力画面を利用する運用にした．

内部処理での訂正は，図 3.10 に示したトランザクション時間のみによる更新になる．図で届出日に 6 月 2 日を，トランザクション時間に 6 月 3 日を指定して検索することにより，6 月 2 日届出のデータとして訂正を反映した履歴を持たない検索結果が得られる．このように，内部処理での訂正では，検索結果に示されるように住民票に訂正履歴は出力されなかった．

一方，業務処理での訂正を行う場合は訂正履歴が必要になる．例えば住民票では，変更を行う職権修正，削除を行う職権削除，追加を行う職権記載があり，訂正の履歴が記載された．図 3.15 に図 3.10 に示した内部処理での訂正を，業務処理での訂正として実施した例を示す．図 3.15 で (1) はトランザクション時間が 6 月 3 日，届出日が 6 月 2 日のスナップショットであり，図 3.10 の (2) に対応する．業務処理での訂正は 6 月 3 日の届出日とな

データベースの状態

6/2の状態　　[Na, Nd):ユーザ定義時間(届出日)

ID	Ta	Td	Va	Vd	Na	Nd	事由	氏名	住所
001-1	6/2	now	6/1	now	6/2	now	出生	時制　太郎	1丁目
002-1	6/2	now	5/31	now	6/2	now	転入	変更　次郎	2丁目
003-1	6/2	now	5/21	now	6/2	now	出生	削除　三郎	3丁目

6/3の状態

ID	Ta	Td	Va	Vd	Na	Nd	事由	氏名	住所	(1)	(2)
001-1	6/2	now	6/1	now	6/2	now	出生	時制　太郎	1丁目	●	●
002-1	6/2	6/3	5/31	now	6/2	now	転入	変更　次郎	2丁目		
002-1	6/3	now	5/31	6/3	6/2	6/3	転入	変更　次郎	2丁目	●	●
002-2	6/3	now	6/3	now	6/3	now	職権修正	変更　次郎	5丁目		●
003-1	6/2	6/3	5/21	now	6/2	now	出生	削除　三郎	3丁目		
003-1	6/3	now	5/21	6/3	6/2	6/3	出生	削除　三郎	3丁目	●	●
003-2	6/3	now	6/3	now	6/3	now	職権削除	削除　三郎	3丁目		●
004-1	6/3	now	6/3	now	6/3	now	職権記載	追加　四郎	4丁目		●

スナップショット

(1) トランザクション時間=6/3, 届出日=6/2

ID	Ta	Td	Va	Vd	Na	Nd	事由	氏名	住所
001-1	6/2	now	6/1	now	6/2	now	出生	時制　太郎	1丁目
002-1	6/3	now	5/31	6/3	6/2	6/3	転入	変更　次郎	2丁目
003-1	6/3	now	5/21	6/3	6/2	6/3	出生	削除　三郎	3丁目

(2) トランザクション時間=6/3, 届出日=6/3以前

ID	Ta	Td	Va	Vd	Na	Nd	事由	氏名	住所
001-1	6/2	now	6/1	now	6/2	now	出生	時制　太郎	1丁目
002-1	6/3	now	5/31	6/3	6/2	6/3	転入	変更　次郎	2丁目
002-2	6/3	now	6/3	now	6/3	now	職権修正	変更　次郎	5丁目
003-1	6/3	now	5/21	6/3	6/2	6/3	出生	削除　三郎	3丁目
003-2	6/3	now	6/3	now	6/3	now	職権削除	削除　三郎	3丁目
004-1	6/3	now	6/3	now	6/3	now	職権記載	追加　四郎	4丁目

図 3.15: 業務処理でのデータ訂正

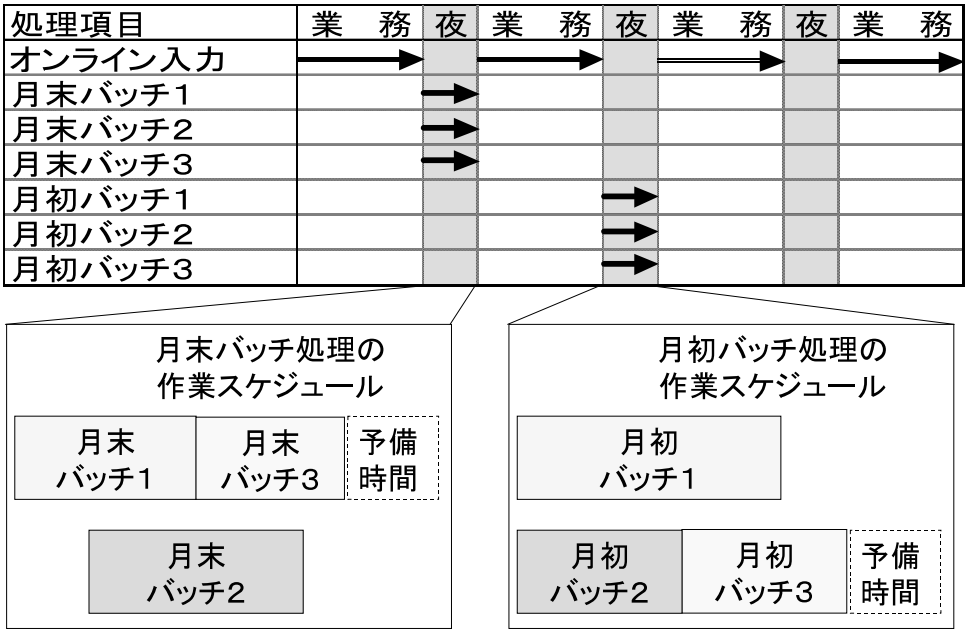


図 3.16: 従来システムのバッチ処理の運用

るため，訂正結果は検索対象にならない．一方，届出日による訂正の履歴が残されるため，図 3.15 の (2) に示す様にトランザクション時間が 6 月 3 日，届出日が 6 月 3 日以前，すなわち，$N_a \leq 6/3$ なる条件を指定した場合には，訂正結果と共に訂正前の履歴も検索対象になる．このように，業務処理での訂正では住民票に訂正履歴が出力された．

3.3.6　バッチ処理の運用

図 3.8 に示す様に，オンライン入力されたデータは各種のバッチ処理によって検索される．検索処理は，検索するデータの時刻の点から，以下の 3 種類に区分された．

■ 業務終了時点の検索処理

日次，あるいは月次で業務終了時点の人口動態，登録台数調べなどの統計が行われ，翌日以降の指定された期日までに結果が報告された．これらは，従来のシステム運用では窓口業務終了後に夜間バッチとして実行されていた．適用システムでは前日末のトランザクション時間のデータを検索することで翌日以降の業務時間内に実行する運用とし，夜間

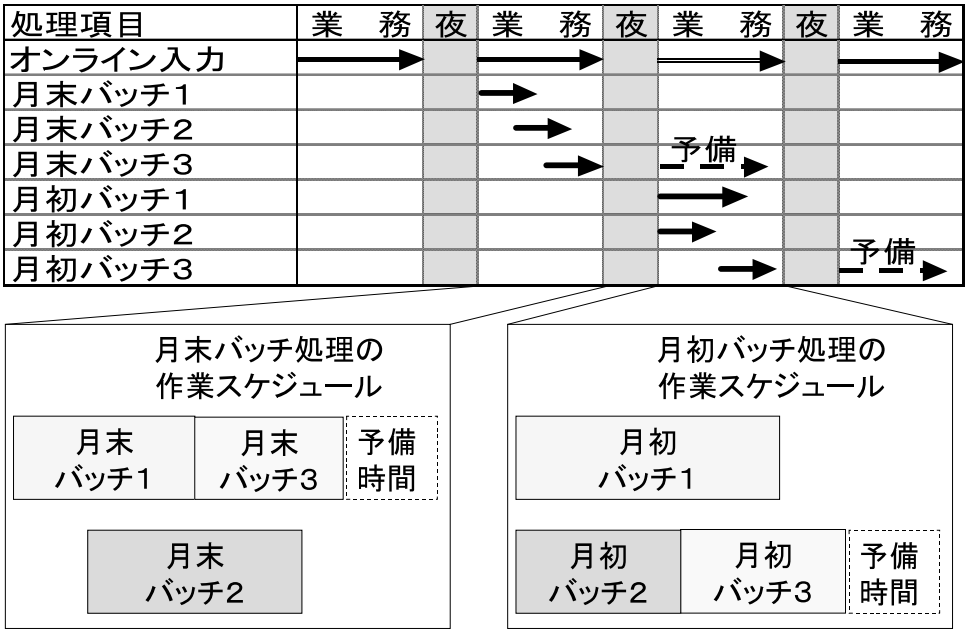


図 3.17: バイテンポラルデータベースによるバッチ処理の運用

バッチを削減した．バッチ処理では仮処理で検索結果のチェックを行い，データの誤りが検出された場合には，必要に応じて図 3.10 に示す様に変更，削除，追加のデータの訂正を行った後でバッチ検索処理，仮処理を再実行した．また，自治体への届出までに時間を要するデータに関しては，図 3.10 の届出日を使用して検索した．

図 3.16 に従来システムの夜間バッチでの月末，月の初日の運用を示す．従来の方式では月末，月の初日という指定期日のデータを使用するバッチ処理は，当日の業務終了後，翌日の業務開始前に実行する必要があった．このため，スケジュールはデータ確認で誤りが検出された場合のデータ訂正，再実行のための予備時間を含め，該当期日の夜間に設定する必要があった．一方，図 3.17 にバイテンポラルデータベースを適用したシステムでの運用を示す．バイテンポラルデータベースにより，データ訂正も含め翌日の業務時間内にバッチ処理が実行可能になった．また，緊急性が低い処理は翌々日以降に回すことが可能になったため，予備時間を別の日に設定できるなど，柔軟なスケジュール設定が可能になった．

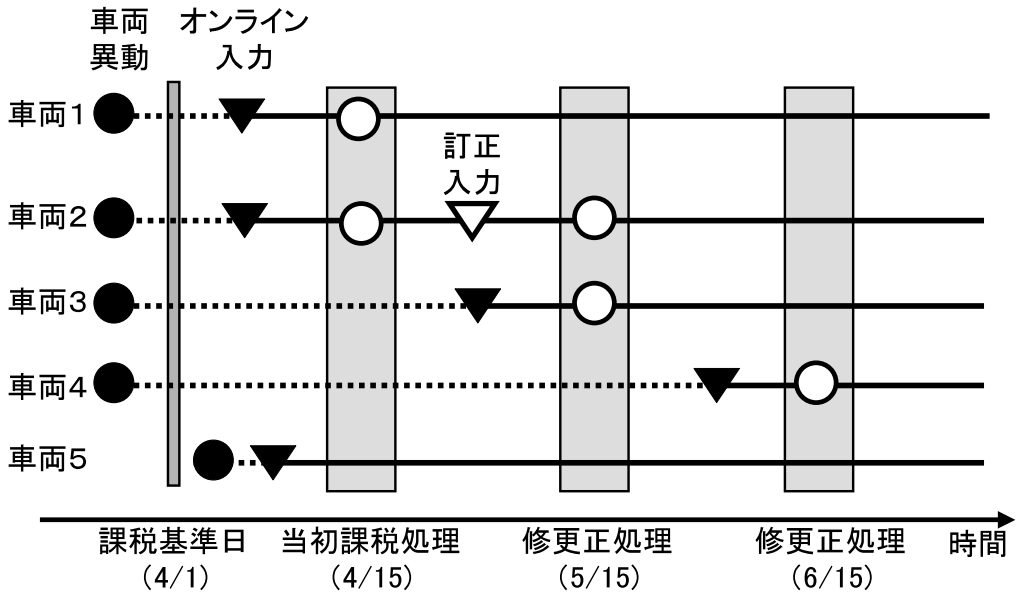


図 3.18: 軽自動車税の当初課税処理・修更正処理のスケジュール

■ 有効時間を指定した検索処理

例えば，課税処理では課税の基準日が設定され，その一定期間後に課税処理が行われた．しかし，課税対象物件の異動届出は時間を要する場合が多く，入力の遅れ，誤りに対する課税の訂正が必要だった．

例として，図 3.18 に軽自動車税の課税処理のスケジュールを示す．図で,「●」は車両異動の発生した期日，すなわち有効時間を示す．また,「▼」はオンライン入力,「▽」は訂正入力,「○」は当初課税あるいは修更生処理のスナップショットのトランザクション時間を示す．軽自動車に対する課税は，毎年度，4 月 1 日を基準日として行われる．実世界の車両の異動である，取得の申告は 15 日以内，廃車，譲渡の申告は 30 日以内とされているが，実際に自治体に申告されるまで，さらに時間を要する場合がある．このため，年度毎に当初課税処理が行われて，以降は定期的に課税の訂正である修更正処理が行われた．図 3.18 では，4 月 15 日を課税処理日とし，以降，1ヶ月毎に修更正処理を行う例を示している．基準日以前の異動結果は該当年度の課税対象となるため，図の車両 1 から車両 4 は入力が行われた直後の当初課税，あるいは修更正の処理対象になる．また，車両 2 のように一旦課税された後，訂正があった場合には，次の修更正の処理対象になる．なお，車両 5 に示されるように，基準日以降の異動は該当年度の課税処理の対象にはならない．

図 3.19 に，有効時間を基準日の 4 月 1 日，トランザクション時間を課税処理日，ある

いは修更正処理日として検索することで，車両の異動を検索した例を示す．図に示されるように，指定した期日までの訂正を反映した，基準日現在の状態を検索できた．

■ 複数のデータ時刻による検索

従来のシステムでは，例えば，定期的に実行する集計処理では，集計処理を実行する都度，結果のデータを累積しておく必要があった．バイテンポラルデータベースを利用したバッチ処理では，図 3.11 に示すように複数の時刻のデータを組み合わせた処理が構築できた．このため，都度，結果を累積せず，最新の処理を実行する際に過去のデータも検索することで，データ管理の運用を容易にした．また，過去のデータを検索する際には，トランザクション時間を最新として訂正を反映したり，過去の処理時刻を指定して訂正を反映しないデータとして検索したりすることで，業務の内容に応じて必要なデータを検索した．

また，バイテンポラルデータベースにより指定した有効時間，およびトランザクション時間のスナップショットによりバッチ処理を行ったが，一部の属性では最新のデータを検索する場合があった．例えば，住民への送付物はバッチ処理の指定期日現在ではなく，それ以降の届出も反映した最新の住所に送付する必要がある．このように，最新の情報が必要な属性で以下の条件を満たすものは，個別にトランザクション時間 now を指定して検索を行い，他の属性は指定した時刻の検索結果とすることが可能になった．

- 該当処理の他の結果に影響を与えない属性
- 検索処理の実行中にデータが更新された場合でも，更新前後のいずれを使用しても問題ない属性

3.4 評価

3.4.1 バッチ検索処理の要件に対する評価

図 3.17 に示すように，従来，夜間に実行していたバッチ処理を翌日以降の業務時間内にオンライン入力と並行して実行できた．すなわち，実際のシステムにおいてもバッチ検索処理の要件の，オンライン入力との並行実行（要件 1）と，スケジュールの柔軟性（要件 3）を満たすことができた．このとき，バッチ処理で誤ったデータが検出された場合に

データベースの状態

ID	Ta	Td	Va	Vd	税額	(1)	(2)	(3)	区分
001-1	1/10	now	1/1	now	1,000	●	●	●	
002-1	1/10	4/20	1/1	now	1,200	●			
002-1	4/20	now	1/1	3/31	0				
002-2	4/20	now	4/1	now	1,600		●	●	変更
003-1	1/10	5/20	1/1	now	5,200	●	●		
003-1	5/20	now	1/1	3/31	0				
003-2	5/20	now	4/1	now	7,200			●	変更
004-1	1/10	4/20	1/1	now	2,500	●			
004-1	4/20	now	1/1	3/31	0				削除
005-1	1/10	5/20	1/10	now	4,000	●	●		削除
006-1	4/20	now	1/1	now	7,200		●	●	追加
007-1	5/20	now	1/10	now	3,000			●	追加

スナップショット

(1) トランザクション時間=4/15，有効時間=4/1

ID	Ta	Td	Va	Vd	Na
001-1	1/10	now	1/1	now	1,000
002-1	1/10	4/20	1/1	now	1,200
003-1	1/10	5/20	1/1	now	5,200
004-1	1/10	4/20	1/1	now	2,500
005-1	1/10	5/20	1/1	now	4,000
				合計	13,900

(2) トランザクション時間=5/15，有効時間=4/1

ID	Ta	Td	Va	Vd	Na
001-1	1/10	now	1/1	now	1,000
002-2	4/20	now	4/1	now	1,600
003-1	1/10	5/20	1/1	now	5,200
005-1	1/10	5/20	1/1	now	4,000
006-1	4/20	now	1/1	now	7,200
				合計	19,000

(3) トランザクション時間=6/15，有効時間=4/1

ID	Ta	Td	Va	Vd	Na
001-1	1/10	now	1/1	now	1,000
002-2	4/20	now	4/1	now	1,600
003-2	5/20	now	4/1	now	7,200
006-1	4/20	now	1/1	now	7,200
007-1	5/20	now	1/1	now	3,000
				合計	20,000

図 3.19: 修更正処理における指定した有効時間での検索

も，通常業務のオンライン入力と並行して，データの訂正および訂正を反映した再処理が可能になった．なお，オンライン入力では業務処理での異動，訂正の他に，トランザクション時間のみを使用して行う内部処理での訂正が行われたが，この両方の更新について一貫性のある検索結果を得ることができた．従って，データ訂正時の再実行（要件 2）も満たすことができた．

こうして，従来は業務終了後に実施していた夜間バッチがなくなったため，バッチ処理をシステム部門から業務主管部門に移管でき，バッチ処理の依頼あるいは帳票の受取りなどの連絡・調整に要する作業を効率化できた．さらに，処理の重複する日には緊急性の低い処理を後日実施するなど，柔軟にバッチ処理のスケジュールを組める様になった．また，実際の業務運用上は，特定の時期に負荷が集中しないように入力されたデータを分割して処理することが必要だった．例えば，毎月の集計であれば 1 日毎に入力データの確認を実施しておくことで処理当日のデータ確認件数や訂正すべき誤り件数を抑えて負荷を削減することが必要になる．バイテンポラルデータベースでは，トランザクション時間によりデータの入力時刻を管理しているため，1 日毎に入力データの確認を行ったり，負荷によっては後日まとめて複数日の入力データの確認を行ったりすることができ，柔軟な運用が可能になった．なお，毎日の集計に関しても，午前，15 時，業務終了の 1 時間前のように時間を区切って入力データの確認を行うという運用が可能になった．

このように，バイテンポラルデータベースを適用することにより，夜間バッチの削減や，柔軟なスケジュールでのシステム運用が可能になり，運用負荷が削減されるという効果が確認できた．例えば，人口 4 万人程度の自治体における夜間バッチ削減の事例では，従来システムにおいて 1 日平均 1.5 時間程度あった残業時間帯のバッチ処理関連作業が削減できた．

なお，図 3.10 に示す住民の異動の様に，実世界のデータは必ずしも即時にシステムへは反映されなかった．この様なデータの統計処理のうち，当日の業務終了時点の結果を翌日に報告する必要があるものは，有効時間ではなくユーザ定義時間である届出日を使用することによって当日のデータの状態を検索できた．ただし，届出日に該当する属性がない場合や，届出がなされたデータを即時にシステムに入力するという運用を行わない業務では，検索対象データを絞りこめないという課題が発生した．この問題に関しては，第 4 章で詳細に議論する．

3.4.2 バイテンポラルデータベースの基幹系システムへの適合性の評価

基幹系システムへのバイテンポラルデータベースの適用結果から，バッチ検索処理の要件の，実際の業務システムへの適合性（要件 4）を評価する．適用システムでは，図 3.19 に示す課税処理の様に長期に渡り訂正データを管理し，異動統計によって時系列に訂正の状況を把握する必要のある業務があった．従来システムの運用では訂正結果を反映した前回の集計結果を検索できないため，前回以降の集計を別途管理し，結果に反映するという作業が必要だった．この様な業務に対しても変更，削除，追加の全ての訂正を反映した検索が可能になり，異動統計として容易に前回と今回の集計，およびその間の異動に関する合計を検索することができた．この結果，データの整合確認が容易になり，誤りが検出しやすくなるという効果があった．なお，この検索処理も他のバッチ検索処理と同様に，オンライン入力中に実行することができた．

図 3.14 に示すように，オンライン入力されたデータは内部で確認を行ってから，業務処理で利用される．住民記録システムなど履歴を管理する業務システムでは，内部での確認段階で検出した誤りの訂正は，住民票などの公的な資料に記載されない．このような訂正は内部処理での訂正として実行された．一方，住民票の発行などの業務処理で利用された後に誤りが検出された場合には，業務的な手続きによる訂正が必要であり，住民票に訂正履歴が記載された．バイテンポラルデータベースにより，内部処理での訂正と，業務処理での訂正の両方を管理することができた．この結果，従来システムでは訂正履歴を残さなかった内部処理での訂正に対しても，データの経緯が容易に把握できるようになり，データに関する問題が発生した場合の追跡調査が容易になった．

また，過去に処理を行った時刻のデータと現在のデータなど，複数の時刻のデータを組み合わせて処理を行ったり，送付物の送り先住所のように一部の属性のみ最新の情報としたりする業務があった．従来システムでは，このような処理に対しては，所定の時刻毎に処理を実行して個々の結果データを作成し，全てのデータが揃った時点で統合するという運用が必要だった．この結果，中間データの管理やスケジュール管理などのシステムの運用管理が煩雑になっていた．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両方の履歴を管理しており，処理の必要性に応じてさまざまな時点のデータを組み合わせて処理を行うことができた．従って，複数の時刻現在のデータを記載する帳票や送付データを，データが揃った時点で直接作成することが可能になり，システムの運用管理を簡易にすることができた．

3.4.3　バイテンポラルデータベースの実装に関する評価

■ 履歴管理に伴うデータ量の評価

バイテンポラルデータベースの実装にあたっては，履歴管理によりデータ量が増大するという課題があった．すなわち，従来のバイテンポラルデータベースの方式では，データ量については図 3.9 の (1) に示す様に，データ変更の際に変更前，変更後の 2 件のデータを追加する必要があった．適用システムでは，この対策として有効時間を図 3.9 の (2) に示すイベント発生時刻により表現することで 1 件の追加に削減できた．このため，有効時間の終了時刻を管理する必要のあるテーブルを除いては，イベント発生時刻による表現を採用した．この結果，業務で必要とする履歴情報に対するデータ量の増加は，内部処理での訂正データ分の増加のみになった．適用システムのテーブルでは，この増加は最大でも年に 20%程度であり，5 年間のライフサイクルで 2 倍以下だった．近年は記憶媒体の単位容量の価格が下がっており，データ量の増大は構築コストの面での問題にはならなかった．

なお，業務上，有効時間の終了時刻を管理する必要のあるテーブルとしては，例えば，以下のテーブルがあった．

- **データに有効期限のあるもの**
  ある住民の異動は，最後には死亡，転出，職権削除というイベントによって完了し，それ以降の有効時間においては該当の住民は自治体には存在しない．しかし，一方で終了のイベントのない情報がある．例えば，税金の口座引き落としのための口座管理に関しては，別の口座への切り替えは新規の口座情報の登録というイベントが発生するが，口座引き落としの停止というイベントを実装しようとすると，仮想的な異動の区分を追加する必要があり，複雑化をまねく．このため，データに有効期間のあるものは終了時刻を管理するのが実装上，効率的だった．

- **開始時刻と終了時刻をセットで扱うもの**
  自治体で管理しているさまざまな資格，例えば，国民健康保険，老人医療，福祉行政に関する各種の受給資格は，必ず「取得」と「喪失」のセットになっており，1 人の住民が同一の資格に関して何度も取得と喪失を行うことがしばしば発生している．例えば，通常は農業に従事し冬季のみ会社員として勤務する住民は，毎年，国民健康保険と健康保険の切り替えを行うため，国民健康保険システムでは同一資格の取得と喪失が繰り返されることになる．このようなデータは検索の際に，ある取得に対応する喪失の検索が複雑になるのを避けるため，実装上は開始時刻と終了時刻の

(a) テーブルのデータ

車両テーブル				
S-ID	Ta	Td	V	J-ID
002-1	1/10	4/20	1/1	100
002-1	4/20	now	1/1	100

住民記録テーブル			
J-ID	Ta	Td	V
100	1/3	5/30	1/1
100	5/30	now	4/15

(b) 単純な結合演算の結果

車両テーブル				住民記録テーブル			
S-ID	Ta	Td	V	J-ID	Ta	Td	Va
002-1	1/10	4/20	1/1	100	1/3	5/30	1/1
002-1	4/20	now	1/1	100	1/3	5/30	1/1
002-1	1/10	4/20	1/1	100	5/30	now	4/15
002-1	4/20	now	1/1	100	5/30	now	4/15

図 3.20: 履歴を持つデータの結合演算における課題

セットで資格取得期間を持たせる構成にした.

■ 検索機能の実装の評価

バイテンポラルデータベースでは，トランザクション時間と有効時間の両方の履歴に対する検索となるため，特にテーブルの結合演算はスナップショットデータベースや，他の時制データベースにおける検索に比較して複雑になる．これは有効時間をイベント発生時刻による表現とした適用システムでも同様であった．例えば，図 3.8 に示す住民記録テーブルと車両テーブルは共に，トランザクション時間と有効時間の履歴を持つ．ここで，図 3.20 に示すように車両テーブルには住民の識別子である $J-ID$ を外部キーとして持たせたが，両者は共に履歴を持つため (b) に示すように履歴に関する直積の検索結果が得られてしまうという問題がある．ここで，両者のトランザクション時間および有効時間は同期していないため，時間の情報は結合演算のキーとしては使用できない.

この問題に対しては，バッチ処理の中間ファイルをデータベースのテーブルであるワークテーブルによって構成した．すなわち，図 3.11 に示すように検索処理でテーブル毎に指定した時刻のスナップショットを抽出し，以降のプログラムでも SQL による検索機能を利用して，段階的にデータを処理していく構成にした．例えばバッチ検索処理の結果としてスナップショットデータベースと同様のスナップショットを抽出し，以降はスナップ

ショットデータベースの場合と同様に処理する構成として，個々のプログラムを単純化できた．この結果，複数のテーブルの結合演算など，複雑な検索を行う場合にも商用リレーショナルデータベースの SQL，および SQL/PMS の機能範囲で構築できた．

なお，ワークテーブルの利用にあたっては，大量のデータを扱う場合にはインデックスを利用すると insert が遅くなるため，insert 後，あるいはインポートによる生成後にインデックスを生成するという配慮が必要だった．

■ オンライン入力の実装の評価

適用システムでは，オンライン入力は楽観的な方法を採用した．自治体システムでは，オンライン入力が届出書の確認などで時間を要することから，データをロックする時間の短い楽観的方法は有効だった．なお，楽観的方法では，データの入力を完了した更新段階になってから他のトランザクションでの更新有無を確認するため，操作を再度やり直すことが必要になる場合がある．この点でも，適用システムでは届出書に基づく入力が多いため，同時に複数の端末から同一の住民に対する更新操作を行うことは，通常の運用では発生しないため問題はなかった．

ただし，楽観的方法の実装にあたっては，バイテンポラルデータベース特有の配慮が必要になった．まず，更新を行う場合には他のトランザクションによる更新中か確認を行い，他のトランザクションが更新中であれば更新を中断する．バイテンポラルデータベースではスナップショットデータベースと異なり，図 3.12 に示すように更新操作は該当する更新前データへの削除時刻の設定のみを行い，更新後のデータは新たに追加される．従って，更新操作では，更新後のデータをロックすることによる排他制御が実行できない．これついては，更新前データを対象として確認フェーズと更新フェーズで更新ロックを行うことで，更新の競合を防止できた．

また，データの追加ではスナップショットデータベースの主キーに加えてトランザクション時間の主キー属性が加わるため，この属性値が異なるデータが二重に登録されるという問題があった．この問題は，図 3.13 に示すように，トランザクション時間属性のうちの削除時刻の属性を主キー属性とすることで解決できた．これは，トランザクション時間を追加する場合には，常に終了時間のインスタンスは now となっているため，追加するタイミング，すなわち追加時刻が異なっても，削除時刻が同一になることによる．

このように，バイテンポラルデータベースにおいても，楽観的方法による同時実行制御を用いて一貫性のあるオンライン入力を行うことができた．

■ 時間の粒度の評価

バイテンポラルデータベースは，時系列に変化するデータを扱うため，時間軸の最小単位である粒度の決定が重要になった．適用システムでは，トランザクション時間の粒度はデータの更新頻度から決定した結果１秒とし，データベースの更新の際に楽観的方法の確認フェーズで更新有無を判定するために使用した．トランザクション時間はユーザには公開しないため，全てのテーブルで同一の粒度とした．

一方，有効時間はユーザの管理項目であるため，その粒度は業務の必要性からテーブル毎に個別に決定された．適用システムの例では時・分，日，月，年・年度があった．また，適用システムでは届出日などのユーザ定義時間は，有効時間を代替する属性として使用したため，有効時間と同様にテーブル毎に個別の粒度が設定された．

このように，適用システムでは時間の粒度は，トランザクション時間は更新頻度の点から１秒に，有効時間は業務要件の点からテーブル毎に設定された．

3.5 考察

3.5.1 バッチ検索処理の要件に関する考察

バイテンポラルデータベースにより実際の基幹系システムにおいても，バッチ検索処理の要件を満たすことが分かった．この結果，オンライン入力中であっても，訂正データを反映した一貫性のある検索結果が得られ，指定期日の処理を後日実行することにより，夜間バッチの削減や柔軟なスケジュール設定によるシステム運用負荷の削減が可能であることが確認できた．

近年は顧客の利便性の点から，オンライン入力を伴う業務システムのサービス時間が延長される傾向にあり，夜間バッチの時間短縮が問題になっている．自治体においても，一部の窓口の業務時間を延長することにより，住民の利便性向上が行われた．さらに，電子申請，電子商取引などの Web システムの進展でノンストップサービスが拡大しており [72, 64, 80, 87]，夜間バッチのようにオンラインを停止して大量データを処理するという運用は，今後，困難になっていくと考えられる．このようなシステムの運用動向に対して，オンライン入力を停止することなく，バッチ検索処理を行う方式は有効であると考えられる．

また，実際の業務運用では実世界の状態が即時にシステムに反映されるとは限らなかっ

た．このため，業務によっては実世界におけるデータの有効時間とは別に，届出日などのユーザ定義時間を管理し，これを有効時間の代替属性として検索処理を行うことで，バッチ検索処理の要件を実現できることが分かった．ここで，ユーザ定義時間は業務面でも使用された．例えば，自治体システムでは住民の年齢計算は，当然ながら有効時間である誕生日により計算される，一方，人口統計表や，選挙の有権者登録は指定期日の状態を即時に把握する必要があり，届出日を基準にして異動が把握された．このように，業務面では両方の時間属性が必要になる場合があり，処理の内容により時間属性が使い分けられた．

ただし，実際の業務においては，実世界のデータが即時にシステムに反映されず，しかも届出日のように有効時間を代替する属性が管理されていない業務があった．このような業務に対しては，バイテンポラルデータベースによるバッチ検索処理が適用できない場合があることが分かった．この問題については，第 4 章で議論する．

3.5.2　バイテンポラルデータベースの基幹系システムへの適合性の考察

基幹系システムをバイテンポラルデータベースにより構築することにより，バッチ検索処理の課題が解決できるだけでなく，データ管理の面での効果があることが分かった．

まず，データ変更管理の点では，特定の有効時間のデータに対し，長期に渡る訂正が発生する業務があった．例えば，図 3.19 に示す課税処理や修更正処理では，課税基準日現在のデータに対し長期に渡る訂正が発生し，通常で５年，違法な場合には７年に及ぶことが法律で定められている．従って，これらの業務では指定した実世界の基準時刻のデータが，システム内で時間の経過に伴いどの様に訂正されてきたかという履歴管理が必要だった．バイテンポラルデータベースは，実世界の基準時刻に対応する有効時間と，システム内での時間経過に対応するトランザクション時間の両方を管理しており，長期に渡るデータの経緯把握や，異動統計の作成などの業務運用での効果があることが確認できた．他の業務分野でも自治体と同様に法律などでデータの保存期間が設定されており，時効などで訂正義務がなくなるまでは誤った処理に関するデータの訂正が発生する．従って，長期に渡ってデータの訂正履歴を管理できる機能は有効であると考えられる．

同じく，実際の業務でのデータの訂正は，業務処理での訂正と内部処理での訂正があった．前者は各種の資料に履歴が残されるが，後者は履歴を残さない．バイテンポラルデータベースでは，業務処理での訂正の他に，内部処理での訂正についても，トランザクション時間を使用することで訂正履歴を管理できた．内部処理での訂正は帳票や画面には現れないが，データベース内部では訂正履歴が蓄積される．また，ユーザが変更することがで

きないため，耐改ざん性の高いデータ変更管理が可能になった．基幹系システムでは不正防止のために耐改ざん性や，高い監査性の維持が必要であり，本機能は有効であると考えられる．

また，データ保管管理の点では複数の期日のデータを統合した処理に効果があることが分かった．定期的に処理を行う業務，例えば月次の決算や，請求処理などでは，前回以降の差分の把握や，今回の結果に併せて前回の結果も表示するということが一般的に行われている．従来の運用では，前回の結果を利用する場合には前回分を別途，保管，管理しておき，必要な時に参照するという運用が必要であるため，各時点の処理結果のデータを保管，管理するというシステムの運用管理面での煩雑さがあった．バイテンポラルデータベースでは，前回結果，およびそれ以降の訂正がデータベースの履歴情報として管理され，例えば，前月末時点と今月末時点のデータを一括して検索することができる．この結果，各時点のデータを保管管理する必要がなくなり，システムの運用管理が容易になった．

3.5.3　バイテンポラルデータベースの実装に関する考察

バイテンポラルデータベースの実装については，履歴管理によるデータ量の増大と，検索処理の複雑化という課題があった．これに対し，実際にバイテンポラルデータベースを基幹系システムに適用し，実装方式の改良により，商用のリレーショナルデータベースによってバイテンポラルデータベースを構築可能であることが分かった．

まず，データ量の増大については，図 3.9 に示すように有効時間をイベント発生時刻による表現にすることで，履歴の量を削減できることが分かった．この結果，適用システムでは，履歴の増加量は 5 年間のライフサイクルで，業務上必要な履歴のみを管理する場合の 2 倍以下になった．また，データ量については，近年の記憶媒体の容量あたりの単価下降を考慮する必要がある．近年では単価下降により，2 倍程度のデータ量の増大が直接的に費用面での問題となることは少ないと考えられる．

ただし，データ量の増大に関しては，検索処理の複雑化と相まって検索の応答性能の劣化を引き起こすことが考えられる．従って，バイテンポラルデータベースでは検索処理の実装方式が重要になる．ここで，バイテンポラルデータベースでは，履歴管理を行うため図 3.20 に示すように結合演算をはじめとして検索処理が複雑化する．この課題に対し，図 3.11 に示すように，バッチ処理の中間ファイルとしてワークテーブルを使用して，単純化した検索プログラムにより段階的にデータを処理するのが効果的であることが分かった．本方式により，商用のリレーショナルデータベース上で，基幹系システムに適用可能

なバイテンポラルデータベースが構築できることが分かった.

3.6 むすび

本章では，バイテンポラルデータベースを基幹系システムに適用し，実際のシステム運用において発生する誤入力や入力遅れなどの訂正についても，オンライン入力中に訂正を反映した一貫性のある検索結果が得られること確認した．この結果，夜間バッチの削減やシステム運用のスケジュールを柔軟に構成できるという，システム運用面での効果を確認できた.

さらに，バイテンポラルデータベースを基幹系システムに適用して得られた知見として，長期に渡るデータ訂正を伴う業務で異動統計などのデータ訂正管理が可能になること，データの訂正を内部処理と業務処理の両面から管理できること，複数の時刻時点のデータを統合した処理が可能になることから，システム運用におけるデータ管理面で効果があることが確認できた.

また，バイテンポラルデータベースの実装にあたっては，履歴管理に伴うデータ量の増大や，検索の複雑化という課題があった．これらの課題に対しては，バッチ処理の中間ファイルをワークテーブルで構成して段階的にデータを処理することで個々の検索処理を単純化し，また，有効時間の表現を時間区間ではなくイベント型にすることで履歴の増加を抑止した．この，方式により商用のリレーショナルデータベースの機能範囲で，基幹系システムが構築可能であることを確認できた.

なお，実際のシステム運用では即時に実世界の状態がデータベースに反映されないという課題があることが分かった．適用システムでは，ユーザ定義時間である自治体への届出日を，有効時間の代替属性として利用することで，多くの場合にはバイテンポラルデータベースの検索機能を利用できた．ただし，このような代替属性が管理されていなかったり，届出データを即時にオンライン入力しない業務では課題が残った．この課題については，第 4 章で詳細に議論し，改良のための方式を提案する.

第4章　データ訂正を反映したバッチ検索方式

4.1　まえがき

第3章に，バイテンポラルデータベースを利用することによって2.2節に示した要件を満足するバッチ検索処理を構築できることを示した．しかし，基幹系システムに適用した結果，実際のシステム運用では実世界の状態が入力されるまでに時間を要する場合があり，指定した有効時間のデータがバッチ検索処理を実行した後で入力され，処理対象から漏れるという課題があることが分かった．このため，バイテンポラルデータベースの適用にあたっては，3.3.2節に示したように入力の遅れるデータは届出日を管理し，届出日によって検索するという改良を行った．しかし，届出日から入力までに遅れが発生する場合や，届出日に該当する属性, すなわち有効時間の代替となる属性がない場合には，この改良でも対応できない．

この課題の対象となるバッチ処理としては，日々，入力されるデータを，例えば，日次や月次のように定期的に処理する運用がある．本章では，入力されたデータを定期的に処理していくバッチ処理の運用事例を示し，このような運用ではバッチ処理毎に締め切り時刻が指定され，締め切り時刻までに入力されたデータを対象として処理が行われることを示す．次に，バイテンポラルデータベースでこのような運用を行う場合の課題を示し，バイテンポラルデータベースの改良として，訂正検索を提案する．これは，締め切り時刻現在のスナップショットにそれ以降に行われたデータ訂正結果を反映するものであり，この方式によりバイテンポラルデータベースの課題が解決できること，訂正検索とバイテン

ポラルデータベースの機能が併用できることを示す.

さらに，基幹系システムに適用し，実際の業務運用においても効果があること，バイテンポラルデータベースを利用したバッチ処理に容易に実装できること，オンライン入力だけでなく一括入力されたデータにも適用できることを確認した.

なお，訂正検索はトランザクション時間のみによる検索であるため，バイテンポラルデータベースだけでなく，トランザクション時間データベースへの適用が可能である．そこで，トランザクション時間データベースへの適用のための拡張についても議論する.

本章は以下のように構成される．4.2 節で訂正検索により解決しようとする時制データベースの課題を述べる．4.3 節で課題を解決するための訂正検索の方式を示し，課題が解決できることを示す．併せて，トランザクション時間データベースへの適用に関する議論を行う．4.4 節では，この訂正検索をバイテンポラルデータベースの改良として 3.3 節の基幹系システムに適用した結果を示し，4.5 節で適用結果について評価する．4.6 節で考察を行い．最後に 4.7 節でまとめを述べる.

4.2 バイテンポラルデータベースの課題

4.2.1 対象とするバッチ処理

第 3 章で，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した結果を評価した．その結果，実際のシステム運用ではデータ入力の遅れる場合があり，その場合にはバイテンポラルデータベースでバッチ検索処理の要件に対する課題が発生することが分かった．データ入力の遅れる業務のバッチ処理は，通常，バッチ処理の都度，締め切り時刻が設定され，締め切り時刻までに入力されたデータを対象に処理が行われる．この場合においても，バッチ検索処理の要件 1，2，3，4 を満たすことが必要である.

このようなバッチ処理の事例として，図 4.1 に会計システムにおける支払業務のデータフローの例を示す．支払依頼入力 (a) で，さまざまな支払依頼がオンライン入力され，承認 (b) を経て支払依頼テーブル (1) に蓄積される．これらの入力・承認は常時行われるため締め切り時刻が設定されて，バッチ検索処理 (2) では前回の締め切り時刻以降，今回の締め切り時刻までに入力されたデータを対象にして検索が行われ，検索結果 (3) が抽出される．次いで，仮処理 (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，バッチ検索処理 (2)，仮処理 (4) が再実行される．(4) での確認完了後に，確定処理 (5) により支払データ (6) の更新が行われる．支払依頼テーブルはバイテン

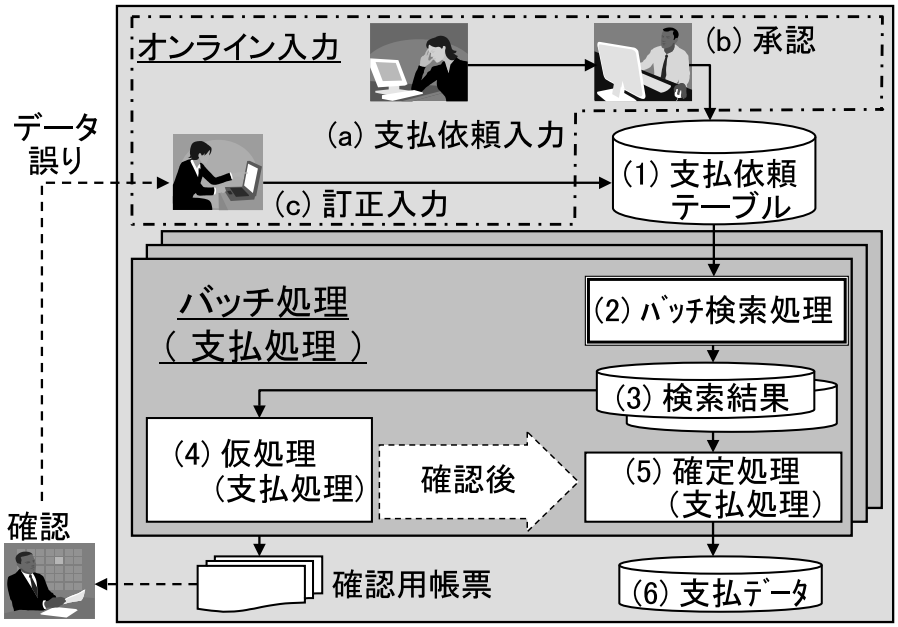


図 4.1: 支払業務のデータフロー

ポラルデータベースであり，リレーションは $R(K, T, V, A)$ で示され，3.2.1 節と同様に以下に示す属性を持つ.

- $K = \{K_1, \cdots, K_m\}$
  トランザクション時間および有効時間を指定したスナップショットで生成されるテーブルの，主キー属性集合.

- $T = \{T_a, T_d\}$
  トランザクション時間の時間区間属性.

- V
  有効時間の時間属性あるいは時区間属性．ここでは，3.3.2 節に示したイベント発生時刻による表現とする.

- $A = \{A_1, \cdots, A_n\}$
  その他の属性集合.

ここで，支払依頼テーブル (1) のリレーションは，$R(ID$，追加時刻，削除時刻，検収日，金額）となり，検収日は物品の購入などで検収を完了した期日になる.

(A) 6/5の支払依頼テーブルのデータ

ID	Ta	Td	V	金額
401	6/5	now	6/1	1,000
402	6/5	now	6/5	2,000
403	6/5	now	6/5	3,000

(B) 6/6の支払依頼テーブルのデータ

ID	Ta	Td	V	金額	TG	T5	T6	備考
401	6/5	now	6/1	1,000	●	●	●	
402	6/5	6/6	6/5	2,000		●		変更前
402	6/6	now	6/4	2,200	●		●	変更後
403	6/5	6/6	6/5	3,000		●		削除
404	6/6	now	6/6	4,000			●	6/6の承認

(C) 検索対象データ

ID	Ta	Td	V	金額	TG	備考
401	6/5	now	6/1	1,000	●	
402	6/6	now	6/4	2,200	●	変更後

図 4.2: 支払依頼テーブルのデータ

図 4.2 に支払依頼テーブルのデータの例を示す．図 4.2 の (A) は 6 月 5 日現在のデータであり，$ID = 401$ のデータは検収日 V が 6 月 1 日にもかかわらず入力が遅れ，6 月 5 日に入力された．図の (B) は 6 月 6 日現在のデータであり，訂正により $ID = 402$ の変更と，$ID = 403$ の削除が行われた．ここで，$ID = 402$ は検収日も変更されている．また，6 月 6 日に $ID = 404$ がオンライン入力された．ここで，6 月 5 日を締め切り時刻とするバッチ検索処理では 6 月 5 日までに入力されたデータの訂正後の状態，すなわち図の (B) の「TG」欄の「●」のデータが検索対象になる．参考までに,「T5」と「T6」に各々，6 月 5 日とは 6 月 6 日のスナップショットを示す．図の (C) に検索対象のデータ，すなわち図の (B) の「TG」のデータを示す.

バイテンポラルデータベースのバッチ検索処理では，このような締め切り時刻による検索は，3.3.2 節に示したように，前回の締め切り時刻となる有効時間を t_{2b}，同じく今回の締め切り時刻を t_{2e} とするとき，検索するトランザクション時間を t_1 のスナップショットとして，式（3.7）により $R_4(t_1, t_{2b}, t_{2e})$ によって表現される．これにより，現在のオンライン入力の影響を受けずに，トランザクション時間 t_1 までの訂正を反映したデータを検索することができる.

ここで，図 4.2 のデータを日次処理で処理する事例，すなわち，有効時間 V のデータを翌日にバッチ処理する運用を示す．この運用では，V の属性値を $r[V]$ とするとき，$ID = 401$

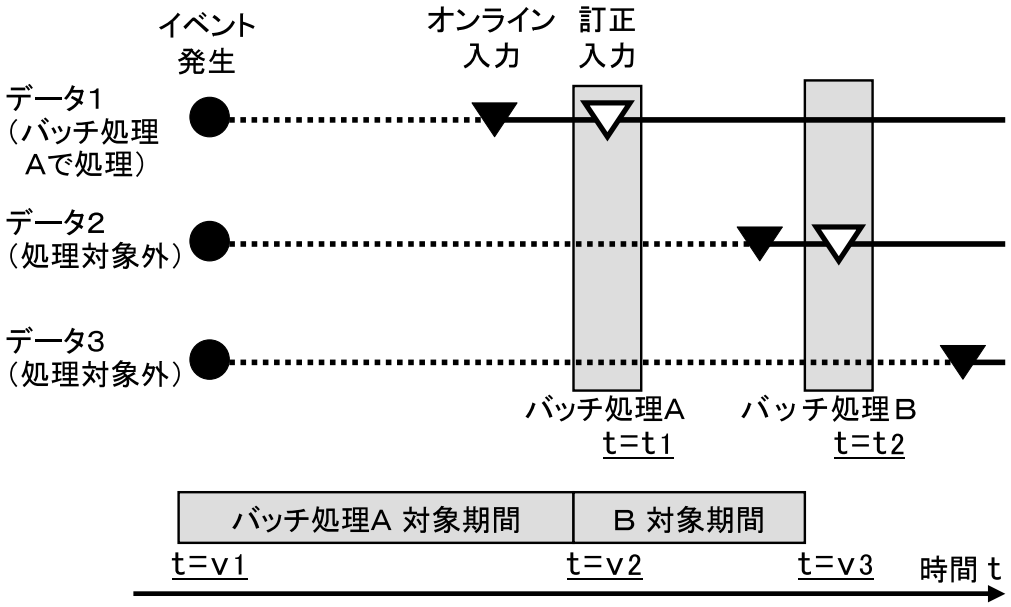


図 4.3: バイテンポラルデータベースにおける定期的なバッチ処理の課題

のデータは $r[V] = 6/1$ のため，6 月 2 日の処理での対象となる．ところが，6 月 2 日には入力されていないため，処理対象から漏れるという課題が発生する．また，$ID = 402$ のデータは 6 月 6 日に $r[V] = 6/5$ から $r[V] = 6/4$ に変更されているため，6 月 5 日の処理からも 6 月 6 日の処理からも漏れるという課題が発生する．このように，バイテンポラルデータベースによる検索処理では，データの入力遅れに伴いバッチ処理で処理漏れが発生するという課題がある．

4.2.2 入力の遅れるデータに関する課題

バイテンポラルデータベースにおける入力の遅れるデータの課題を，詳細に議論する．定期的なバッチ処理はある一定期間のデータを対象に，処理が行われる．図 4.3 に，バイテンポラルデータベースにおける例を示す．図で，イベント発生の時刻「●」は有効時間であり，オンライン入力の時刻「▼」，訂正入力の時刻「▽」はトランザクション時間になる．バッチ処理 A の対象となる時間区間を $[v_1, v_2)$，バッチ処理 A におけるバッチ検索処理のスナップショットのトランザクション時刻を t_1 とするとき，対象となるデータは，式（3.7）により $R_4(t_1, v_1, v_2)$ で表現される．

従って，図 4.3 のデータ 1 は訂正後のデータが検索される．しかし，入力の遅れるデー

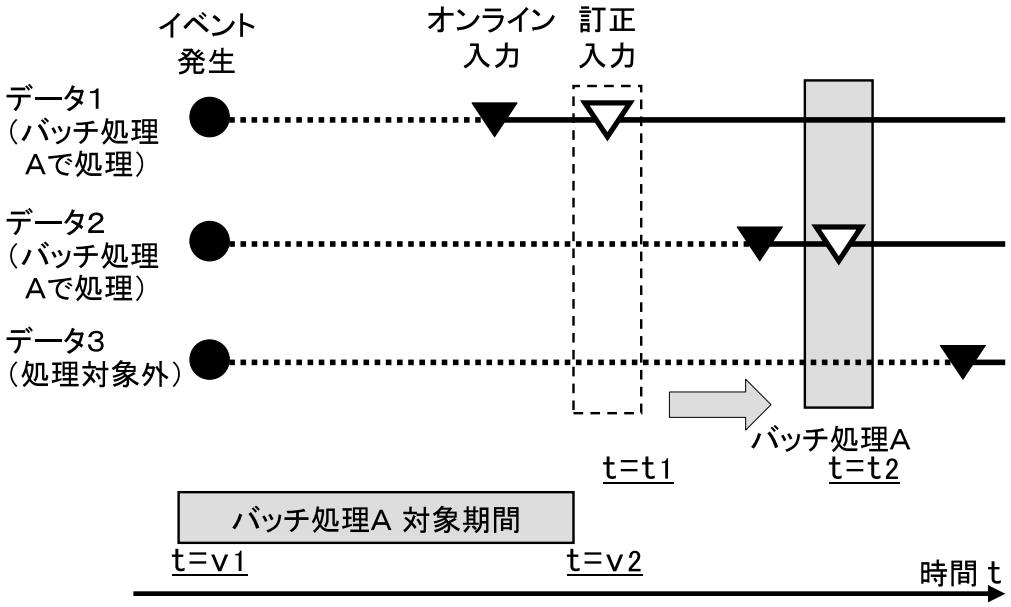


図 4.4: バッチ処理の実行時期を遅らせることによる改善

タ 2，データ 3 は，トランザクション時間 $t = t_1$ 時点では入力されていないためバッチ検索処理の検索対象とならない．また，次の回の定期的なバッチ処理 B については，対象となる有効時間の時間区間がバッチ処理 A の対象時間区間終了後の $[v_2, v_3)$ になるため，データ 2，データ 3 はこの対象とはならない．従って，データ 2，データ 3 はバッチ処理の対象から漏れてしまうという課題がある．なお，バッチ処理 B の対象時間区間を，バッチ処理 A の対象時間区間を含めた $[v_1, v_3)$ とすると，バッチ処理 B におけるバッチ検索処理のスナップショットのトランザクション時刻 $t = t_2$ 時点ではデータ 1 もデータベースに存在しているため，データ 1 が再度処理されてしまう．

同様に，図 4.2 の $ID = 402$ の検収日の変更は，図 4.3 のデータ 1 の訂正入力において，イベント発生の時刻をバッチ処理 A 対象期間以前に変更することになる．この場合，データ 1 はバッチ処理 A の対象外になる．ところが，バッチ処理 A の，前の回のバッチ処理は既に完了しているため，データ 1 はバッチ処理の対象から漏れてしまうという課題がある．

これに対し，実際のシステム運用ではしばしば，図 4.4 に示すバッチ処理の実行時期を遅らせるという運用が取られる．図に示すように，バッチ処理 A の実行時期を $t = t_1$ から $t = t_2$ まで遅らせることで，データ 2 のように，この間に入力されたデータを処理対象とすることができる．この運用は，データの入力期限が決まっており，その間に入力され

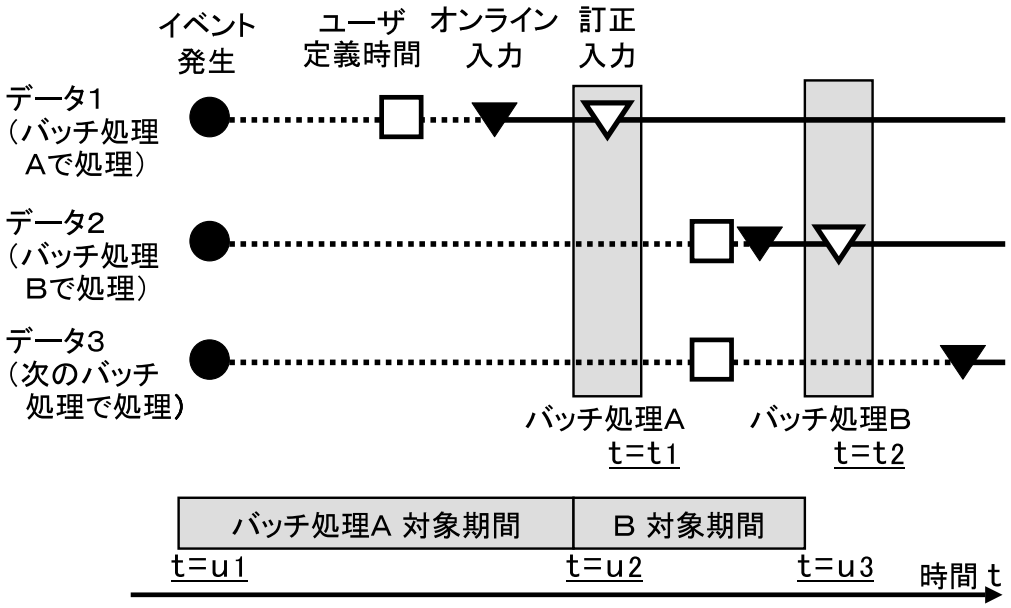


図 4.5: ユーザ定義時間を使用した検索による改善

ないデータは無効になるような業務については有効である．例えば，返品期限の決まっている商品の販売においては，販売した日を有効時間，返品要求の入力された日をトランザクション時間として，バッチ処理による返品状況を処理することが可能である．

しかし，実際の業務では必ずしも有効期限を設定できない．例えば，3.3.1 節の自治体システムの事例では，住民の異動や課税対象物件の異動は，異動日から届出や申請までの期間が規定されているが，正しい住民の台帳作成や課税処理を行うためには届出や申請が遅れたデータも扱う必要がある．

これに対し，上記の自治体システムでは，住民の異動はバイテンポラルデータベースのユーザ定義時間である，届出日によって検索する運用にした．図 4.5 に，ユーザ定義時間を基準とする場合のスケジュールを示す．ここで，バッチ処理 A はユーザ定義時間区間 $[u_1, u_2)$ を，バッチ処理 B は $[u_2, u_3)$ を対象とする．ユーザ定義時間を使用することにより，データ 2 のように入力の遅れたデータが次回のバッチ処理 B で処理される．

しかし，データ 3 のようにユーザ定義時間からさらに入力の遅れる場合には，処理の漏れが発生するという課題がある．さらに，4.2.1 節に示す会計システムにおける支払処理のように届出という運用を取ることのできない業務では，この方式が取れないという課題がある．

4.2.3　データの直接訂正における課題

実際の運用では，このようなデータに対し時制データベースの手続きによらずに，データベースを直接訂正するという処理がやむをえず行われることがある．例えば，図 4.2(B) で $ID = 402$ の有効時間 $r[V] = 6/5$ を $r[V] = 6/4$ に修正するが，この訂正を時制データベースの手続きによらず，データベースを直接訂正する方法がある．この方法では，トランザクション時間の時区間は $[6/5, now)$ のままであるため，6 月 5 日のバッチ処理を再処理することで検索することができる．しかし，この方法では式（3.7）のスナップショットが検索のタイミングにより変化するため，他のバッチ検索処理で該当テーブルを検索中の場合には，その一貫性が維持できなくなるという課題がある．また，データベースの更新履歴が残らないため，トランザクション時間データベースの利点である耐改ざん性や，監査性が維持できなくなるという課題がある．

また，他の方法としては，図 4.1 の検索結果 (3) を直接訂正するという方法がある．この方法は，該当のバッチ処理では正しい結果を得ることができる．しかし，基幹系システムでは各テーブルはさまざまな処理で検索される．従って，この方法では支払依頼テーブル (1) を検索する他のバッチ処理の結果と整合しなくなるという課題がある．さらに，バッチ処理で，バッチ検索処理から再処理を行う必要が発生した場合には，再度，同じ訂正を行う必要があるという課題がある．

4.3　データ訂正を反映した検索方式

バイテンポラルデータベースのバッチ検索処理において，入力が遅れたデータが検索対象から漏れるという課題を解決するため，トランザクション時間のみによって検索する訂正検索を提案する．訂正検索では，検索する際に指定したトランザクション時間以降に該当データが訂正されているかを調べ，訂正されていればそれを検索結果に反映する．

4.3.1　訂正検索

■ 訂正検索の検索方式

訂正検索はトランザクション時間によるスナップショットの検索時刻 t_1 に加え，$t_1 < t_2$ なる訂正検索時刻 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_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検索方式である．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

$$S = S_1 \cup S_2 \tag{4.1}$$

で表現される．

- S_1：t_1 から t_2 までに変更・削除されないデータ
  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．該当データは時刻 t_1 と t_2 に存在するため，以下で表現される．

$$S_1 = \{s | s \in R_1(t_1) \wedge s \in R_1(t_2)\} \tag{4.2}$$

  ここで，R_1 は式 (3.2) で示されるスナップショットのリレーションである．

- 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集合値 $r[K]$ と $s[K]$ で対応付けられるため，該当データは式 (4.3) で表現される．なお，t_2 までに削除されたデータは検索対象外になる．

$$S_2 = \{s | s \notin R_1(t_1) \wedge s \in R_1(t_2) \wedge \exists r \in R_1(t_1), r[K] = s[K]\} \tag{4.3}$$

ここで，訂正検索で検索される各々のデータは，時刻 t_2 のスナップショットのデータ $R_1(t_2)$ の部分集合であり，通常のトランザクションによる更新結果になる．従って，個々のデータの一貫性は維持されている．また，$R_1(t_2)$ は過去の時刻のスナップショットであるため，データの更新中であっても，その影響を受けない．すなわち，訂正検索により，データ更新中であってもその影響を受けず，変更，削除の訂正を反映した一貫性のある検索結果を得ることができる．

訂正検索の例を図 4.6 に示す．検索時刻 t_1，訂正検索時刻 t_2 の訂正検索結果は，t_1 のスナップショットに t_2 までに発生した訂正，すなわち，$ID = 302$ の変更，$ID = 303$ の削除を反映したものになり，新たに入力された $ID = 304$ は反映されない．

■ バッチ検索処理要件に対する適合性

訂正検索が 2.2 節に示した以下のバッチ検索処理の要件を満たすことを示す．まず，オンライン入力との並行実行（要件 1）については，訂正検索結果は過去のトランザクショ

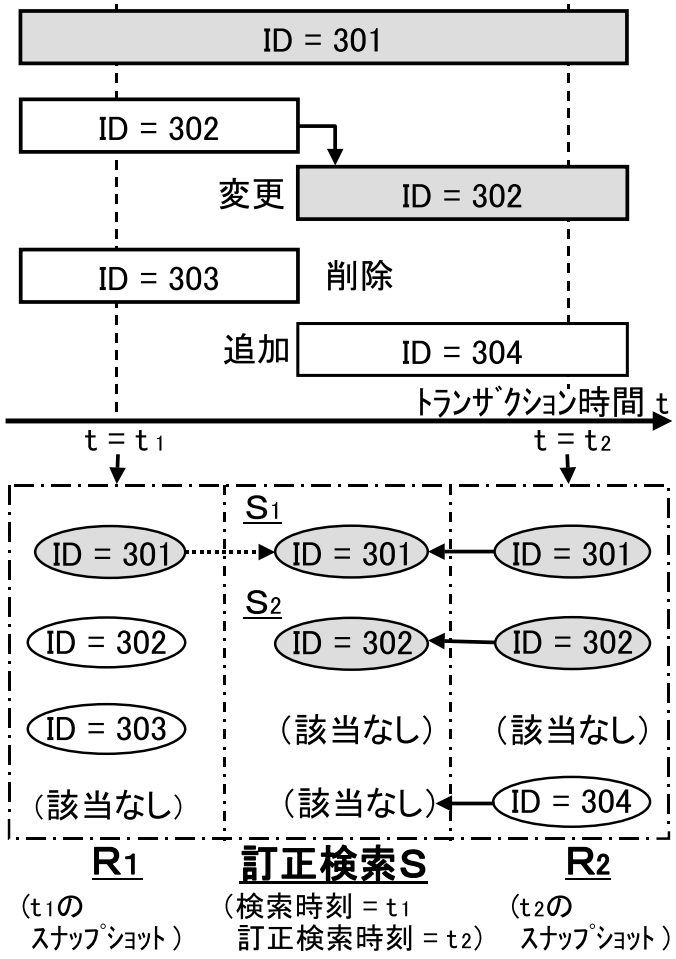


図 4.6: 訂正検索の検索方式

ン時間 t_2 のスナップショットの部分集合になる．従って，2.4.1 節に示すように現在実行されているオンライン入力の影響を受けないため，並行実行が可能である．

データ訂正時の再実行（要件 2）については，図 4.6 に示すように訂正検索の結果は，検索時刻 $t = t_1$ のスナップショットに訂正検索時刻 $t = t_2$ までに行われた訂正を反映したものになる．また，検索時刻 $t = t_1$ で既に入力されていたデータ，およびその訂正データのみが検索対象になるため，$t = t_1$ 以降に新たに入力されたデータは検索対象にならない．従って，データ訂正を行った場合には，訂正結果のみを反映した再検索が可能になる．

図 4.7 に，図 4.2 に示した支払依頼テーブルのトランザクション時間に関する履歴を示す．トランザクション時間 $t = 6/5$ のスナップショットでは訂正前のデータが，$t = 6/6$ では 6 月 6 日の入力が検索されるため，スナップショットでは 6 月 5 日入力分の訂正後の状態が検索できない．これに対し，検索時刻を 6 月 5 日，訂正検索時刻を 6 月 6 日とした

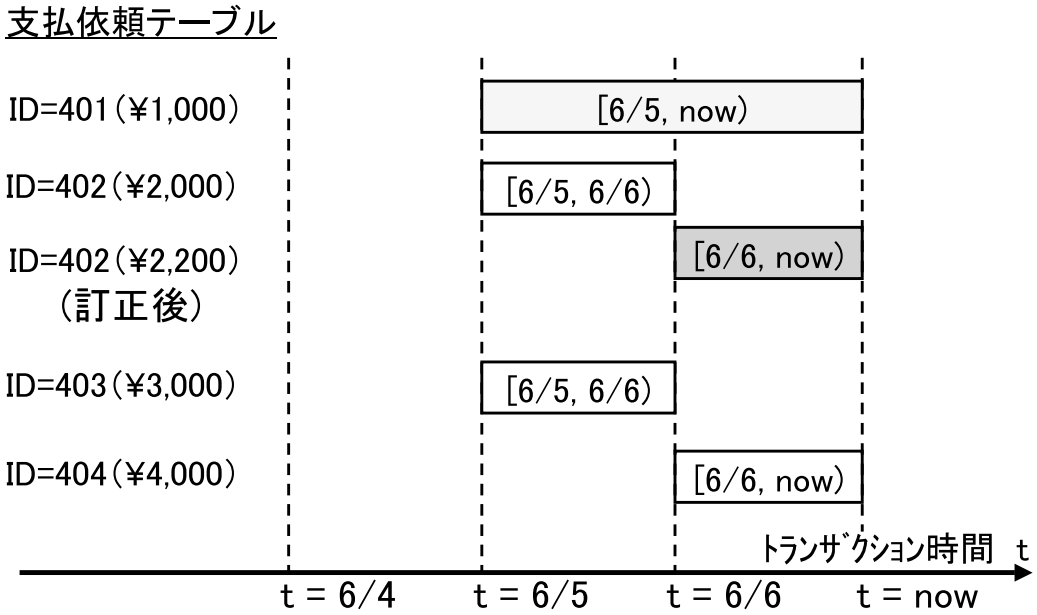


図 4.7: データ訂正後のバッチ検索処理再実行の事例

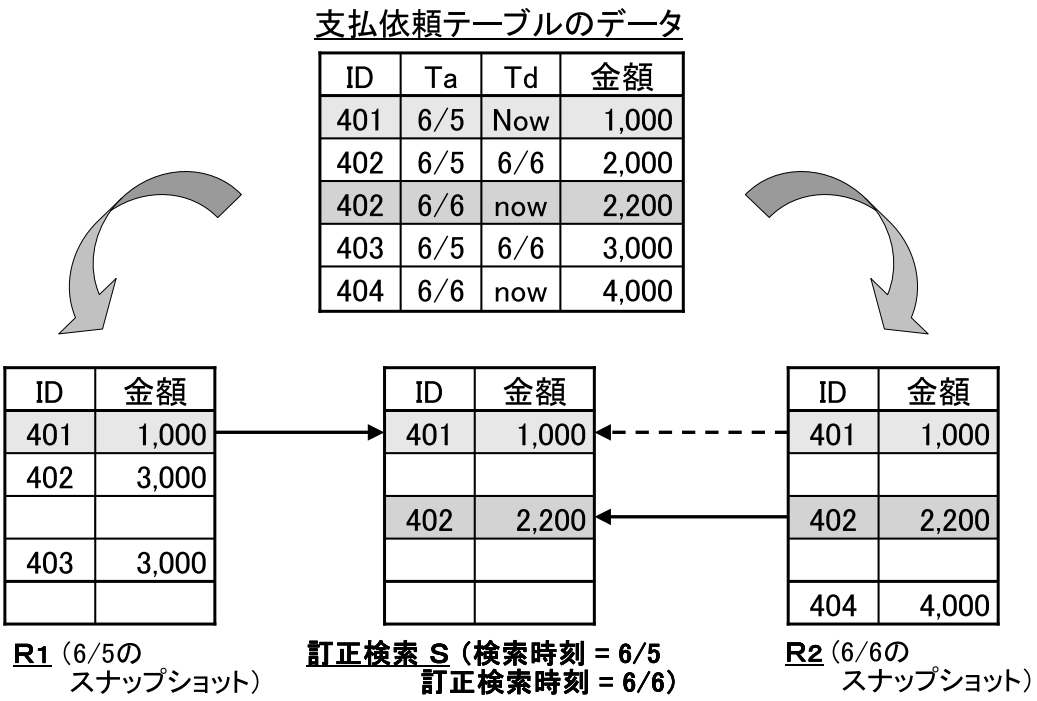


図 4.8: 訂正検索によるデータ訂正後の再実行

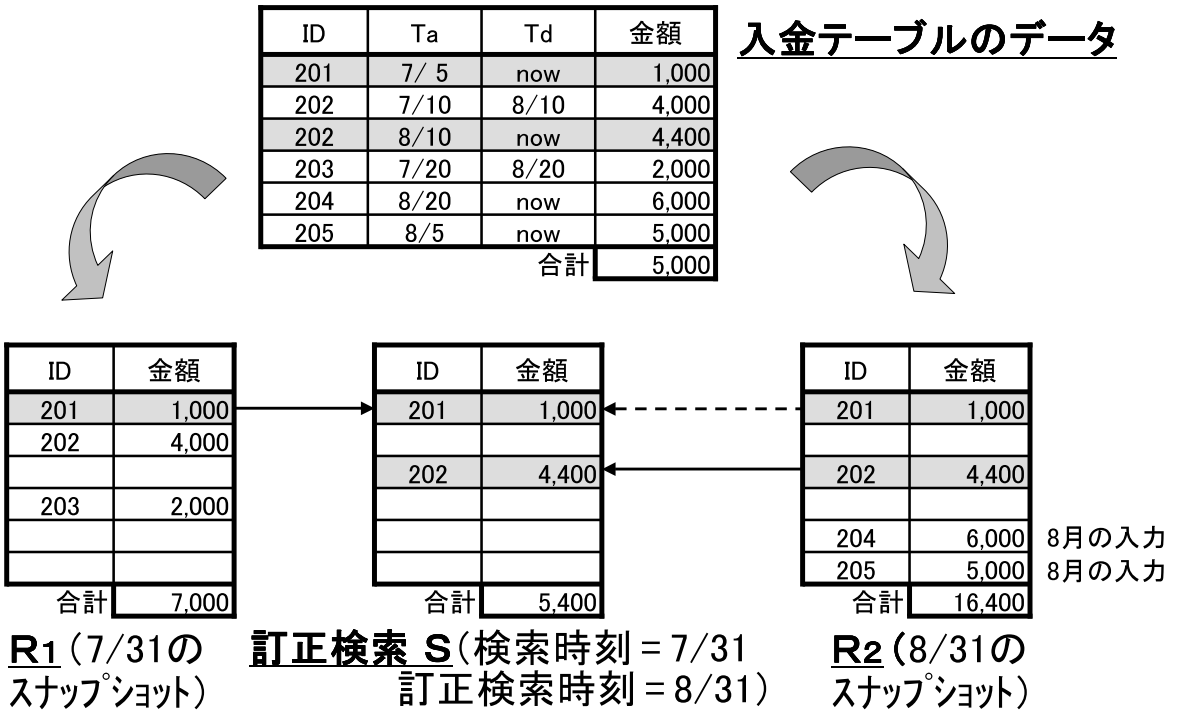


図 4.9: 訂正検索によるスケジュールの柔軟性

訂正検索による検索結果を，図 4.8 に示す．訂正検索では，図 4.8 の 6 月 5 日のスナップショットに対し，6 月 6 日に行われた訂正である $ID = 402$ の変更，$ID = 403$ の削除が反映される．また，$ID = 404$ のデータは 6 月 6 日の入力であるため，訂正検索の対象にならない．この結果，データ更新中であっても，6 月 5 日の入力データに対して変更，削除の訂正を反映した検索結果が得られる．

スケジュールの柔軟性（要件 3）については，過去の任意の検索時刻 $t = t_1$ のスナップショットに対し，訂正検索時刻 $t = t_2$ までに行われた訂正を反映した検索結果が反映されることから，特定時刻現在のバッチ処理を任意の時刻に行うことが可能になる．従って，特定の時刻にバッチ検索処理を行う必要がないため，スケジュールの柔軟性が確保できる．

スケジュールの柔軟性に関しては，図 2.14 に複数の期日の処理を一括して行う場合の事例を示した．図 4.9 に，図 2.14 のデータに対して，検索時刻を 7 月 31 日，訂正検索時刻を 8 月 31 日とした訂正検索結果を示す．7 月 31 日のスナップショットに，8 月 31 日までに行われた $ID = 202$ の変更と，$ID = 203$ の削除が反映され，その結果，8 月 31 日の合計 16,400 円と 7 月 31 日の合計 5,400 円の差異は，8 月の入力 11,000 円，すなわち $ID = 204$ の 6,000 円と $ID = 205$ の 5,000 円の和になり，7 月末と 8 月末の合計，および 8 月の入力

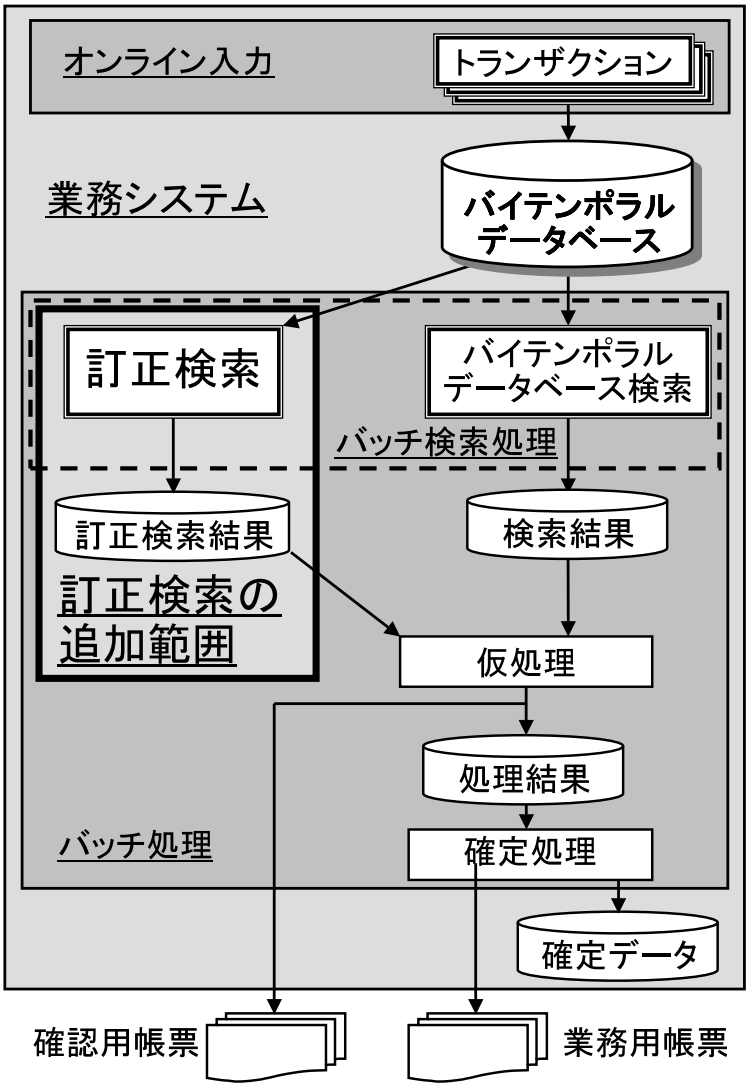


図 4.10: 訂正検索機能を追加した業務システムの構成

が整合する．このように，訂正を反映した 7 月末の集計を 8 月末の集計と同時に行うことができる．すなわち，所定の時刻以降の任意の時刻にバッチ検索処理を行うことで柔軟なスケジューリングが可能になる．

なお，バイテンポラルデータベースでは図 3.5 に示すように，有効時間を使用することにより $ID = 204$ を 7 月入金，$ID = 205$ を 8 月入金とした例を示した．訂正検索では，データが入力された時刻を基準とするため，これらは共に 8 月入力分として 8 月に集計に含まれる．

また，実際の業務システムへの適合性（要件 4）については，訂正検索はバイテンポラルデータベースの検索機能の改良であり，実際の業務システムにおいては，第 3 章のバイ

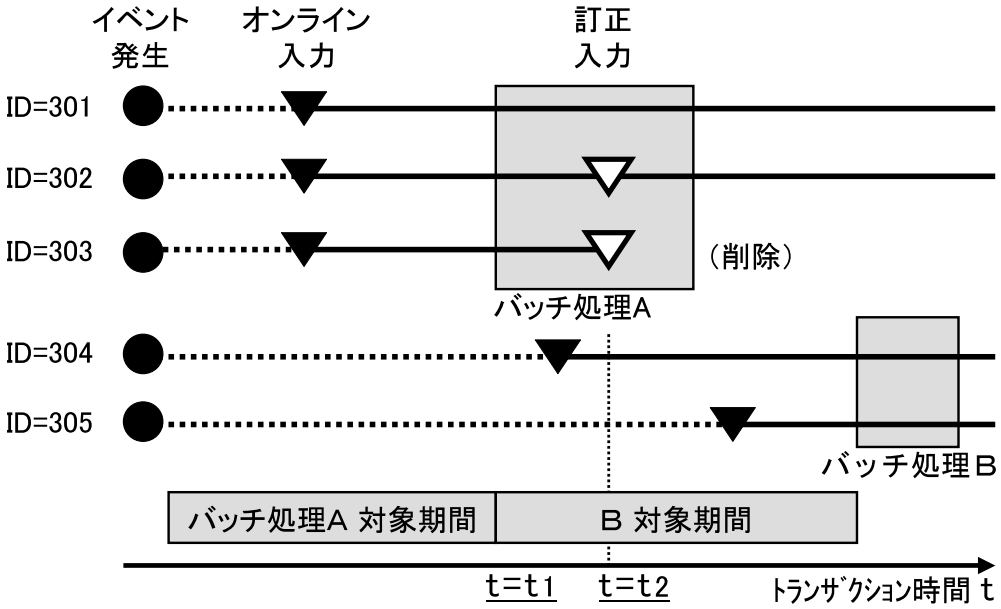


図 4.11: オンライン入力時刻とバッチ処理の関係

テンポラルデータベースの構成に付加した検索機能として実現できる．図 2.1 に示すように，データベースを利用した業務システムは，トランザクションによるオンライン入力機能とバッチ処理によって構成され，バッチ処理はバッチ検索処理，仮処理，確定処理から構成される．図 4.10 に，バイテンポラルデータベースを利用した業務システムおいて訂正検索を追加した構成を示す．訂正検索の追加によってデータベースのリレーションに変更はないため，オンライン入力，および従来のバイテンポラルデータベース検索はそのまま使用できる．また，訂正検索はバッチ検索処理の一部として組み込まれるため，訂正検索の結果はワークテーブルである訂正検索結果テーブルを経由して仮処理に渡される．従って，訂正検索結果テーブルの構成を，従来の検索結果テーブルと同様としておくことで，仮処理，確定処理も従来のバイテンポラルデータベースのバッチ処理の機能と同様に構成できる．このように，バイテンポラルデータベースの機能と併用できるため，業務システムへの適合性も維持される．

■ 入力の遅れるデータに対する効果

訂正検索により，4.2.2 節に示した，入力の遅れるデータに関する課題が解決できることを示す．最初に，バッチ処理の対象から漏れるデータがないことを示す．オンライン入

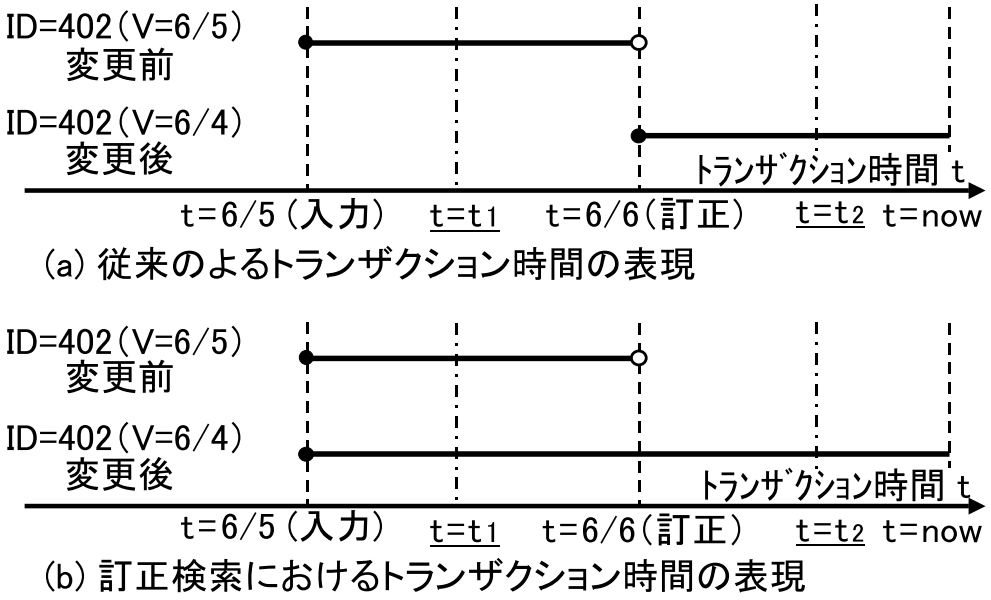


図 4.12: 訂正検索におけるトランザクション時間の表現

力時刻とバッチ処理の関係を図 4.11 に示す．図で，$t = t_1$ が検索時刻，すなわちバッチ処理 A の締め切り時刻であり，$t = t_2$ が訂正検索時刻，すなわちバッチ処理 A で誤りのあるデータ $ID = 302$，$ID = 303$ の訂正完了後の時刻となる．このように，バッチ処理の締め切り時刻 $t = t_1$ までに入力されたデータ $ID = 301$，$ID = 302$，$ID = 303$ が処理対象になり，訂正を反映したデータがバッチ検索処理で検索される．また，$t = t_1$ よりも後で入力されたデータ $ID = 304$，$ID = 305$ はバッチ処理 A の処理対象にはならず，次の回の処理であるバッチ処理 B で処理される．

このように，訂正検索ではトランザクション時間によりデータを検索しており，有効時間を使用しない．従って，各々のデータにはオンライン入力時刻を対象期間とするバッチ処理によって処理されるため，オンライン入力が遅れてもバッチ処理の対象外になるデータは発生しない．また，図 4.2 の $ID = 002$ に示す支払日の変更に対しても，有効時間は使用しないため処理漏れや重複処理は発生しない．さらに，訂正検索では検索の際に訂正を反映するため，4.2.3 節に示した，データを直接訂正する場合の課題は発生しない．

4.3.2 訂正検索の実装

訂正検索を，バイテンポラルデータベース上の検索機能として実装した．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 4.12(a) に示すように，変

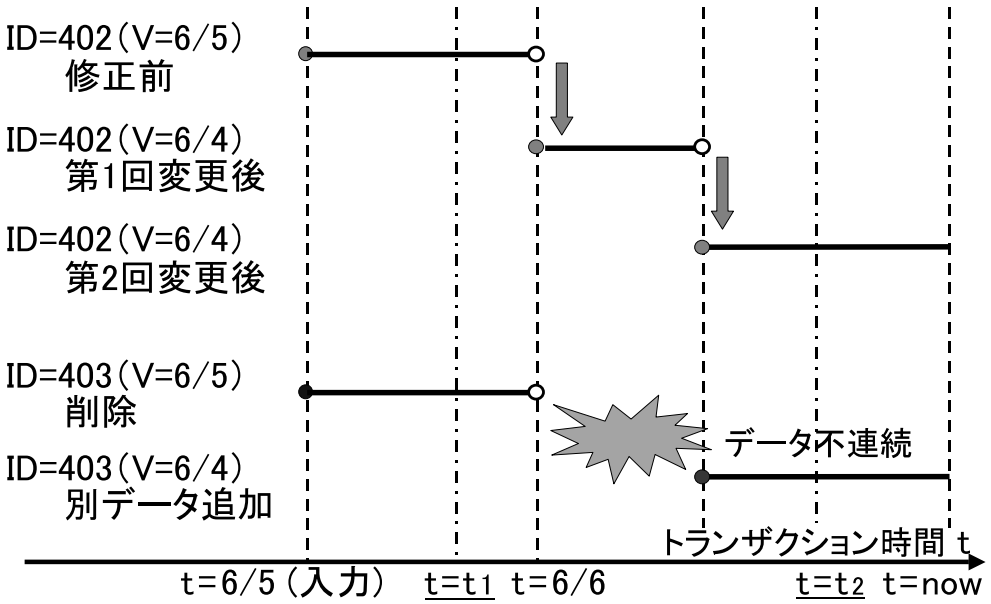


図 4.13: 訂正前後のデータの対応付け

更後のデータは変更時刻を追加時刻とした形式で表現された．訂正検索の実装では訂正前後のデータの対応付けを確認する必要があり，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 4.12(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とした．

ここで，訂正前後のデータの対応付けが必要になる理由を図 4.13 に示す．訂正検索では検索時刻 $t = t_1$ で入力されていたデータを検索対象とする．ところが，図 4.13 のデータ $ID = 403$ は一旦，削除されており，訂正検索時刻 $t = t_2$ 時点のデータは，同一の ID を利用して新たに追加されたデータになっている．従って，図 4.12(a) の表現では，例えば，図 4.13 のデータ $ID = 402$ の 3 件のデータについて $t = t_1$ から $t = t_2$ までの間でデータが途切れていないこと，すなわち各削除時刻に対し，同じ時刻を追加時刻とするデータがあることを確認する必要がある．

なお，従来の表現では追加時刻が主キーの属性になるが，この表現では削除時刻を主キーの属性にする必要がある．この実装では，3.3.4 節に示したとおり楽観的方法によるデータベースの更新を行うため，削除時刻を主キー属性としており，そのまま訂正検索にも適用できた．

この表現により，訂正検索を容易に実装できた．例えば，図 4.8 の訂正検索は，検索時

(a) 納品テーブル

ID	Ta	Td	V(納品日)	請求金額
399	5/29	now	5/29	5,000
400	5/31	now	5/30	5,000
401	6/2	now	6/1	1,000
402	6/4	now	6/4	2,200
403	6/5	6/6	6/5	3,000
403	6/6	now	6/6	3,300
404	6/6	now	6/6	4,000
405	6/6	now	6/6	5,000

(b) 支払依頼テーブル

ID	Ta	Td	V(検収日)	支払金額
399	5/31	now	5/29	5,000
400	6/2	now	5/30	5,000
401	6/5	now	6/1	1,000
402	6/5	6/6	6/5	2,000
402	6/6	now	6/4	2,200
403	6/5	6/6	6/5	3,000
404	6/6	now	6/6	4,000

図 4.14: 納品テーブルと支払依頼テーブルの構成

刻を t_1，訂正検索時刻を t_2 として，以下の SQL で検索できた．

$$select\ ID, Ammount from \text{支払依頼テーブル}\ where \\ r[T_a] \leq t_1\ and\ t_2 < r[T_d] \tag{4.4}$$

なお，この表現では，例えば，図 4.8 の $ID = 402$ の変更後の追加時刻 T_a 値は，変更の行われた 6 月 6 日ではなく，最初にオンライン入力された 6 月 5 日になる．なお，t_2 以降に訂正を行った場合には，同一 ID で複数のデータが検索されるため，検索されたデータの中から ID 毎に t_2 時点で有効なデータ，すなわち削除時刻 $s[T_d]$ が最小のデータを選択する．

次に，バッチ処理における実装について示す．図 4.10 に示したように，訂正検索はバッチ検索処理における機能の追加，すなわち検索ジョブの追加として実装できる．実装の事例として，納品情報と，これに対応して支払依頼情報を管理するテーブルの事例を示す．納品を受けた場合には，検収作業として品質条件，数量，仕様にあっていることを確認し，問題がなければ翌月末に支払を行う．この業務のための，納品テーブルと支払依頼テーブルの構成を図 4.14 に示す．図の納品テーブルでは，納品日を有効時間とし，これと追加時刻，削除時刻によりバイテンポラルデータベースのテーブルが構成される．また，支払依頼テーブルは図 4.8 に同じである．

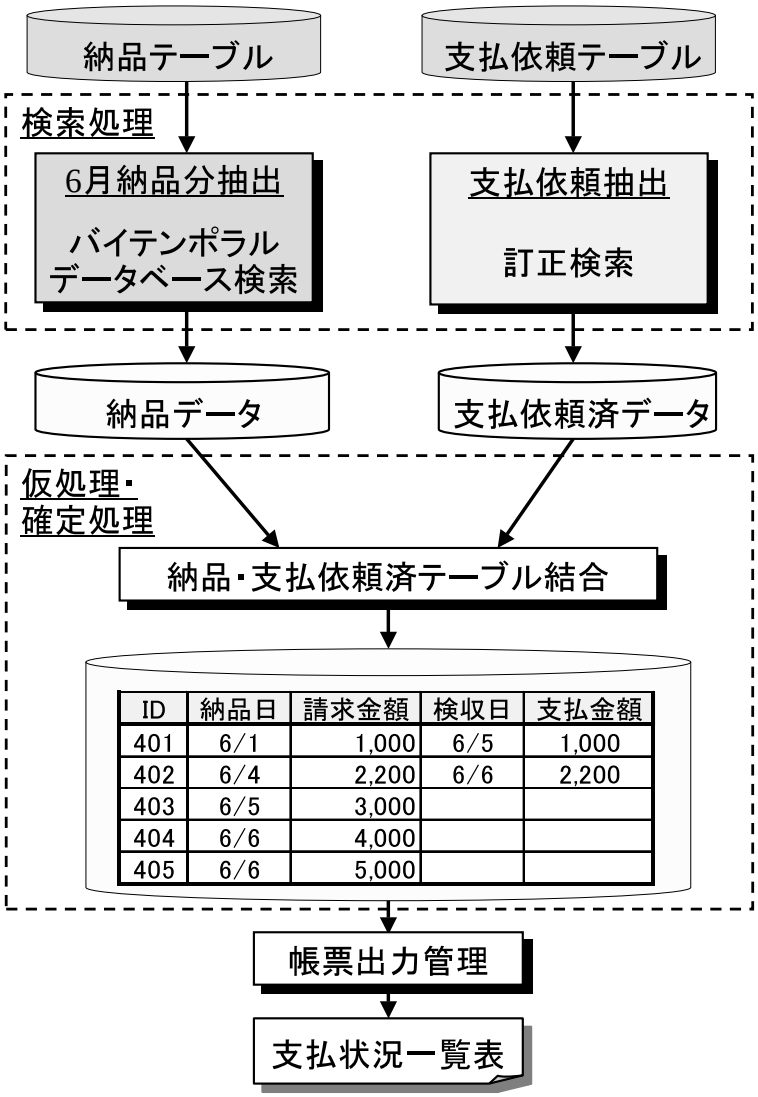


図 4.15: 訂正検索の実装事例

図 4.15 に，6 月 6 日時点の 6 月納品分に対する，支払依頼の処理状況を帳票出力する事例を示す．納品テーブルは承認がないため，即日，入力が行われ，バイテンポラルデータベースとして検索できる．ただし，訂正により入力漏れの追加や，変更が発生する．各々のテーブルの検索では，以下の条件によって 6 月 6 日時点のデータを検索する．

- **6 月納品分抽出**　6 月 6 日時点の 6 月納品分データを抽出
  - 検索方法：バイテンポラルデータベースの検索
  - 有効時間（納品日）：6 月
  - トランザクション時間：6 月 6 日

- **支払依頼抽出**　6月6日時点の6月5日までにオンライン入力されたデータを抽出
  - 検索方法：訂正検索
  - 検索時刻：6月5日
  - 訂正検索時刻：6月6日

抽出結果は中間ファイルに出力されるが，3.3.3節に示したように本実装では中間ファイルはワークテーブルで構成される．従って，仮処理，確定処理では，検索結果の2つのテーブルを，IDと納品日をキーとして結合演算を行うことにより帳票出力用のワークテーブルを作成できる．このように，訂正検索はジョブの追加として実装できる．

4.3.3　トランザクション時間データベースへの適用のための拡張

■ トランザクション時間データベースへの適用に関する課題

訂正検索はバイテンポラルデータベースの改良として位置づけられ，図4.15に示すようにバイテンポラルデータベースと併せて使用することにより，第3章に示すバイテンポラルデータベースの検索機能も併せて使用することができる．ここで，訂正検索はトランザクション時間のみを使用しているため，トランザクション時間データベースにも適用することが可能である．

しかし，訂正検索は締め切り時刻までに入力されたデータを対象とする訂正を前提としており，データの追加による訂正は想定していない．再検索におけるバイテンポラルデータベースの訂正と，訂正検索の訂正を比較する．バイテンポラルデータベースでは図3.4に示すように変更，削除，追加の訂正が可能であり，例えば，図3.19の修更生処理の事例に示すように実際の業務でも必要になった．一方，訂正検索では図4.6に示すように変更と削除の訂正のみが可能である．従って，トランザクション時間データベースにおいて，データ訂正後の再実行（要件2）を満たすためには，追加の訂正を可能にする必要がある．

3.3節の適用システムの事例では，追加は図3.19に示す次の2種類があった．第一はキーとなる情報の誤りによる訂正であり，図3.19のデータ$ID = 005 - 1$が誤っており，5月20日に削除されるとともに，訂正後のデータ$ID = 007 - 1$が追加された．第二は入力漏れであり，同じくデータ$ID = 006 - 1$が該当する．バイテンポラルデータベースでは有効時間の開始時刻V_aにより本来，追加されるべき時刻が管理されており，図3.19に示すようにトランザクション時間に訂正後の6月15日を指定することで検索対象にでき

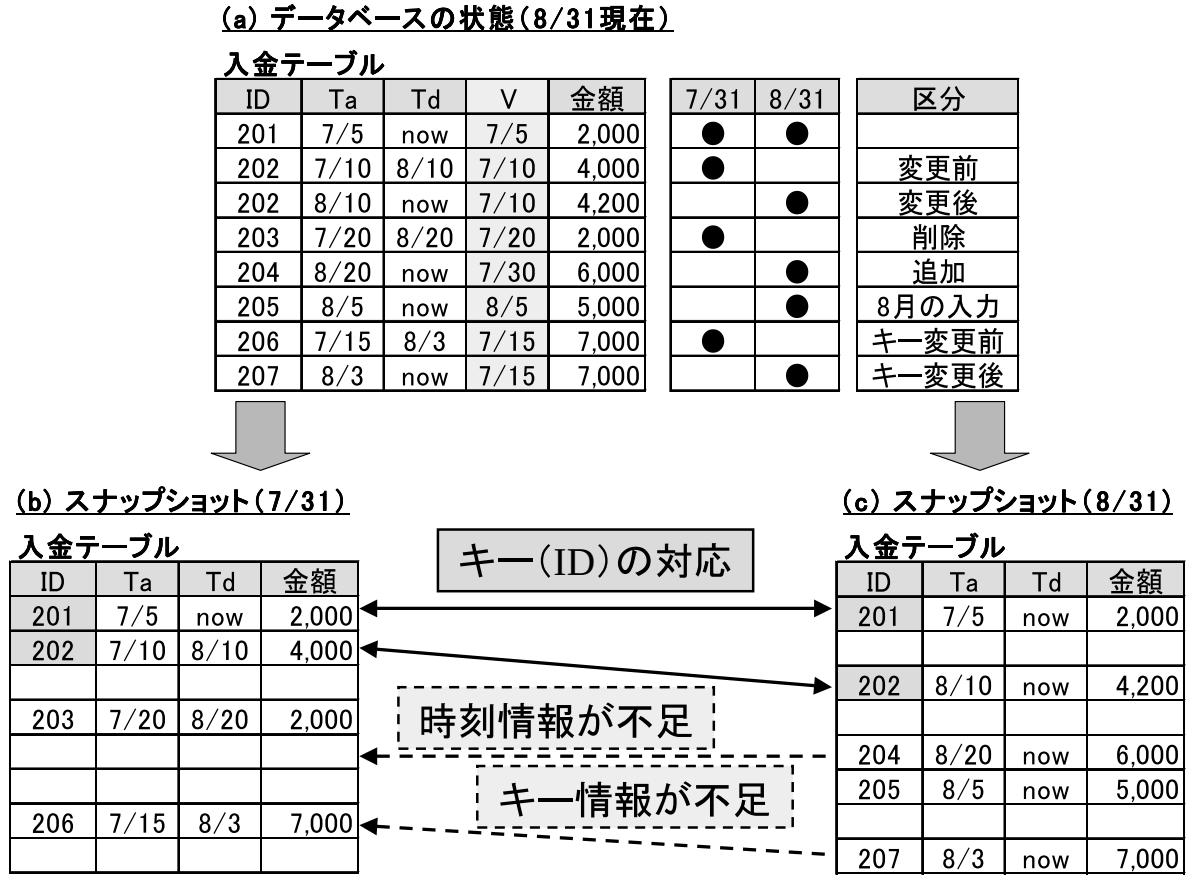


図 4.16: 訂正前後のスナップショットにおけるデータの対応

る．一方で，トランザクション時間データベースでは有効時間が管理されていないため，これを補う属性が必要になる．

上記の 2 種類の場合を含め，検索時刻 t_1 のスナップショットに対し，訂正検索時刻 t_2 のスナップショットまでに行われた訂正を反映するための，双方のスナップショットの対応を図 4.16 に示す．図で，訂正のない $ID = 201$，変更の $ID = 202$，削除の $ID = 203$ および t_2 以降の入力 $ID = 205$ は図 4.6 に示したものであり，訂正検索で対応可能である．一方で，追加の $ID = 204$ およびキー変更の $ID = 207$ は，訂正検索の対象にならない．本節では，トランザクション時間データベースに適用するために，これらの追加の訂正に対する訂正検索の拡張について提案する．

■ データベースの構成

トランザクション時間データベースを拡張し，主キー変更の訂正と，追加の訂正を訂正検索の対象にするための属性を付加する．

- **識別子**
  識別子は，主キー変更が発生した場合に，変更の前後でデータを対応付けるために使用する．

- **訂正追加時刻**
  データが本来追加されるべきだったトランザクション時間を格納する．この情報は，ユーザが必要に応じて入力し，入力がない場合には追加時刻と同一時刻として処理する．これは，時制データベース上はユーザ定義時間に該当する．

  ただし，この属性の目的は，検索時刻現在で該当データが本来，登録されていたかを識別するものであり，必ずしも時間属性である必要はない．

このデータベースのテーブルのリレーション R は

$$R(K, T, D, T_c, A)$$

で表現される．各々の属性を以下に示す．

- $K = \{K_1, \cdots, K_m\}$
  トランザクション時間を指定したスナップショットで生成されるテーブルの，主キー属性集合．

- $T = \{T_a, T_d\}$
  トランザクション時間の時間区間属性．

- D
  識別子の属性．

- T_c
  訂正追加時刻の属性．

- $A = \{A_1, \cdots, A_n\}$
  その他の属性集合．

例えば，図 4.8 の支払依頼テーブルに識別子と，訂正追加時刻を付加したリレーションは，以下になる．

入金 (ID，追加時刻，削除時刻，識別子，訂正追加時刻，入金額)

ここで，識別子には，最初に登録されて時の ID を使用できる．

■ 検索方法の拡張

訂正検索方式を拡張し，式 (4.1) のリレーション S を拡張した訂正検索を

$$\tilde{S} = S_1 \cup \tilde{S_2} \cup \tilde{S_3} \tag{4.5}$$

で表現される．ここで，S_1 は式 (4.2) に同じく，t_1 から t_2 までに変更・削除されないデータであり，$\tilde{S_2}$， $\tilde{S_3}$ は 以下に示すデータである．

- **$\tilde{S_2}$：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．変更前と変更後のデータは識別子の属性集合値 $r[D]$ と $s[D]$ によって対応付けられる．すなわち，変更の場合には t_1 と t_2 の各々の時刻のスナップショットに，主キー属性集合値の一致する別のデータが存在することになる．ここで，t_1 と t_2 の間で削除されたデータは訂正検索の対象にはならない．

$$\tilde{S_2} = \{s | s \notin R_1(t_1) \wedge s \in R_1(t_2) \wedge \exists r \in R_1(t_1), r[D] = s[D]\} \tag{4.6}$$

- **$\tilde{S_3}$：t_1 から t_2 までに追加されたデータ**
  追加されたデータのうち，訂正追加時刻 T_c が検索時刻 t_1 以前のものは，訂正により追加されたデータであるため，訂正検索の対象にする．新たに追加されたものであるため，同一の主キー属性集合値を持つデータが $R_1(t_1)$ に存在しないことが条件になる．該当データは以下で表現される．ここで，T_c が t_1 よりも後である場合には訂正検索の対象にはならない．

$$\tilde{S_3} = \{s | s \notin R_1 \wedge s \in R_2 \wedge s[T_c] \leq t_1 \wedge \forall r \in R_1, r[D] \neq s[D]\} \tag{4.7}$$

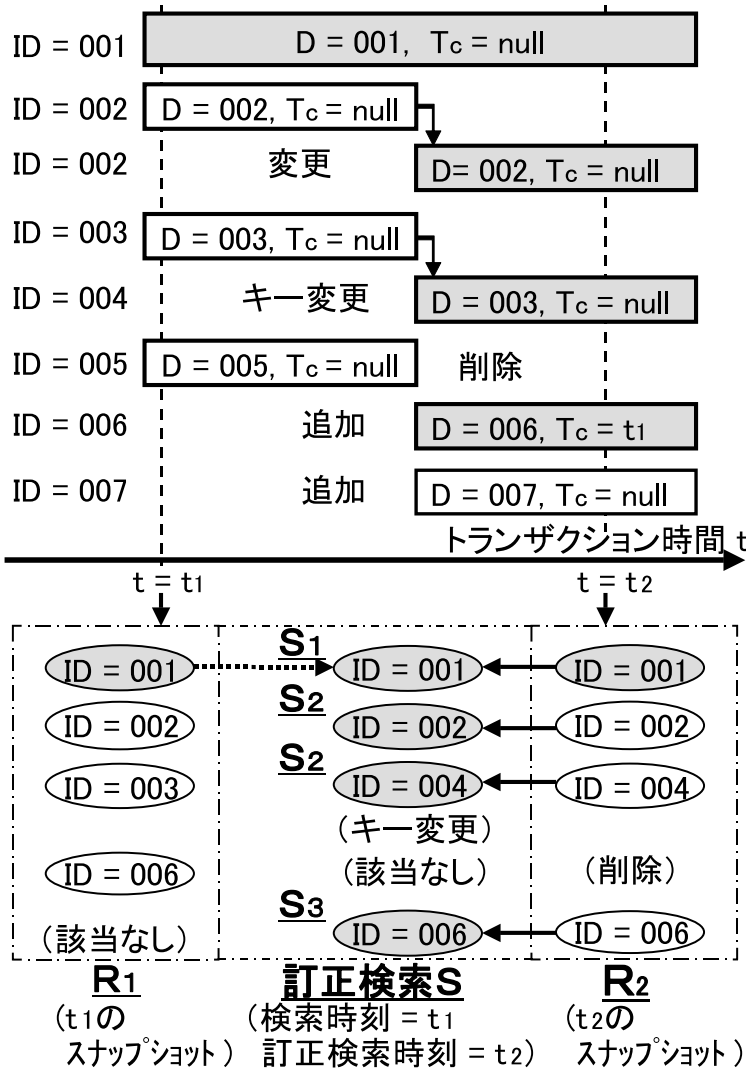


図 4.17: トランザクション時間データベースにおける訂正検索の拡張

トランザクション時間データベースへの適用のために拡張した訂正検索の検索方式を，図 4.17 に示す．検索時刻 t_1，訂正検索時刻 t_2 の訂正検索結果は，t_1 のスナップショットに t_2 までの間に発生した追加，変更，削除の結果を反映したスナップショットになる．

ここで，拡張訂正検索 $\tilde{S}$ で検索される各々のデータは，S と同様に時刻 t_2 のスナップショットのデータ $R_1(t_2)$ の部分集合であり，通常のトランザクションによる更新結果になる．従って，個々のデータの一貫性は維持されている．また，$R_1(t_2)$ は過去の時刻のスナップショットであるため，データの更新中であっても，その影響を受けない．すなわち，訂正検索により，データ更新中であってもその影響を受けず，追加，変更，削除の全ての訂正を反映した一貫性のある検索結果を得ることができる．

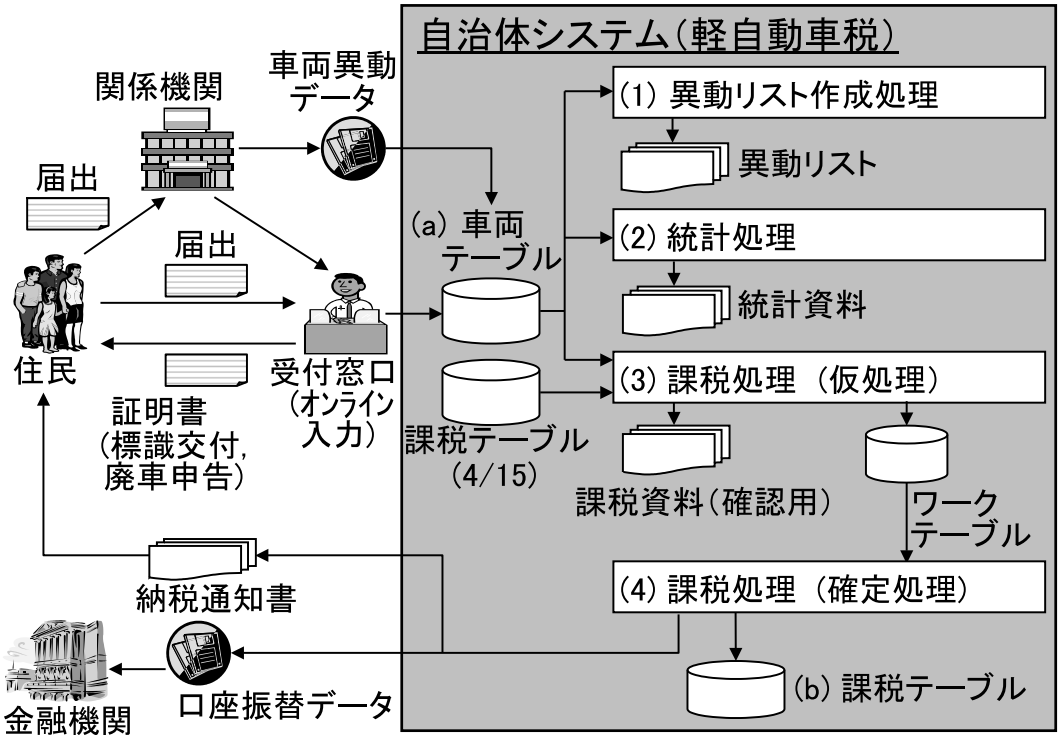


図 4.18: 軽自動車税修更正業務のデータフロー

4.4 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た結果を示す．自治体システムの業務構成は，3.3.1 節に同じである．各業務ではオンライン入力したデータを，日次，月次などの一定期間毎にバッチ処理する．また，データベースも 3.3.2 節と同様に商用のリレーショナルデータベースを使用して構築し，テーブル毎に業務での必要性に応じて追加時刻，削除時刻，および訂正検索機能を付加し，実世界の履歴を管理するテーブルについては有効時間の属性を追加した．適用システムにおける訂正検索のバッチ処理の運用事例として，図 3.6 の税関連システムの軽自動車税における修更正の事例を示す．

4.4.1 軽自動車税修更生業務の構成

軽自動車税は，住民が所有する軽自動車に課税されるものであり，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当初の課税処理が行われる．その後，免除申請，申告遅れなどに対する税額の修更正が月次で行われる．

(a) 4/15の車両テーブルのデータ

ID	Ta	Td	Va	車両	住民	事由
001-1	1/10	now	1/1	001	登録太郎	登録
002-1	1/10	now	1/1	002	廃車次郎	登録
003-1	1/10	now	1/1	003	譲渡三郎	登録

(b) 5/16の車両テーブルのデータ

ID	Ta	Td	Va	車両	住民	事由	(A)	(1)	(2)	備考
001-1	1/10	now	1/1	001	登録太郎	登録	●	●		
002-1	1/10	now	1/1	002	廃車次郎	登録	●	●		
002-2	4/20	5/16	3/31	002	廃車次郎	廃車				廃車(訂正前)
002-2	4/20	now	4/1	002	廃車次郎	廃車		●	●	廃車(訂正後)
003-1	1/10	now	1/1	003	譲渡三郎	登録	●	●		
003-2	4/20	5/16	3/10	003	購入四郎	譲渡				譲渡(→取消)
004-1	4/20	now	4/1	004	新規五郎	登録		●	●	登録
005-1	5/16	now	4/1	005	追加六郎	登録				登録

図 4.19: 異動リスト作成処理における訂正検索

図 4.18 に軽自動車税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期限は取得が 15 日以内，廃車，譲渡が 30 日以内で，自治体の窓口の他に，軽自動車協会，陸運局，商店（以下では一括して，関係機関と記載）で受付けられて自治体に送付される．従って，申告および送付に時間を要するため，実世界の異動は即時に車両テーブル (a) には反映されない．このため，修更正は設定された締め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．また，自治体の窓口では，窓口での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．なお，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．一括入力の場合は，入力データの追加時刻は媒体毎に同一時刻にした．

以下に，図 4.18 の修更生業務の各処理における，バッチ検索処理を示す．以下では，当初の課税処理時刻を 4 月 15 日とし，該当の修更生処理については，締め切り時刻を 5 月 15 日，バッチ処理時刻を 5 月 16 日としている．

ID	Ta	Td	Va	車両	住民	事由	(A)	(B)	(1)	(2)
001-1	1/10	5/16	1/1	001	登録太郎	登録	●			
002-1	1/10	now	1/1	002	廃車次郎	登録	●	●	●	
002-2	4/20	5/16	3/31	002	廃車次郎	廃車				
002-2	4/20	now	4/1	002	廃車次郎	廃車			●	●
003-1	1/10	5/16	1/1	003	譲渡三郎	登録	●			
003-1	1/10	now	1/10	003	譲渡三郎	登録		●	●	
003-2	3/15	5/16	4/1	003	購入四郎	譲渡				
004-1	4/20	now	4/1	004	新規五郎	登録			●	●
005-1	5/16	now	4/1	005	追加六郎	登録				

図 4.20: 統計処理における訂正検索

4.4.2 異動リスト作成処理

図 4.18 の異動リスト作成処理 (1) では申告データが正しいことを確認するための異動リストを出力する．図 4.19 (a) に 4 月 15 日現在，(b) に 5 月 16 日のバッチ処理に伴う訂正を反映した車両テーブルの例を示す．軽自動車の異動は 1 日単位で把握されるため，有効時間の単位は 1 日にした．なお，トランザクション時間は月日のみ示している．

軽自動車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．申告による異動は実世界の履歴として管理されるため ID は変更される．一方，訂正は実世界の履歴ではないため ID は変更されない．異動により $ID = 002-2$，$003-2$，$004-1$，$005-1$ が追加され，5 月 16 日に訂正により $ID = 002-2$ の変更，$ID = 003-2$ の削除が行われた．

異動リスト作成処理では，追加時刻 T_a が 4 月 15 日から 5 月 15 日までの，申告による異動を検索する．図 4.19 (b) の (A) に 4 月 15 日のスナップショット，(1) 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，(2) に (1) のうち 4 月 15 日以降の追加分すなわち異動リスト，の該当データを「●」で示す．(2) では 5 月 16 日の $ID = 002-2$, $003-2$ の訂正が反映された異動が検索され，かつ 5 月 16 日の異動 $ID = 005-1$ は反映されない．すなわち，訂正を反映した 5 月 15 日現在の異動データが検索できた．

4.4.3 統計処理

図 4.18 の統計処理 (2) では，5 月 15 日現在の登録車両台数の統計表を出力し，4 月 15 日の統計表との差異が異動リストと整合することを確認する．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 4.20 に，5 月 16 日の訂正により図 4.19 の $ID = 001 - 1$ の削除，$ID = 003 - 1$ の変更を行った例を示す．(A)，(1)，(2) は図 4.19 と同様であり，(B) 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 (A) と 5 月 15 日の訂正検索結果 (1) の差異は，異動リスト (2) と整合しない．しかし，4 月 15 日の訂正検索結果 (B) により，(B) と (1) の差異 $ID = 002 - 2, 004 - 1$ を異動リスト (2) に整合させることができた．

4.4.4 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 4.18 の仮処理 (3) による確認を行った上で，確定処理 (4) で課税データベース (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引き落しデータ媒体作成を実行した．

図 4.20 で，修更正の対象は 4 月 15 日 現在の状態 (A) と，5 月 15 日の訂正検索結果 (1) に差異のあるデータになる．この例では，車両 001 は課税取消しによる還付，002 は 4 月 1 日に廃車されているため還付，003 は登録日の訂正のみで修更正には影響がなく，004 は申告遅れによる追加徴収が行われた．なお，車両 005 は 5 月 16 日入力のため，次回の修更正の対象になる．

4.5 評価

4.5.1 入力の遅れるデータに関する評価

4.2.2 節でバイテンポラルデータベースでは，オンライン入力データを定期的に処理するためのバッチ検索処理で，データの入力が遅れる場合にはデータ訂正時の再実行（要件 2）に課題があることを示した．4.4 節の適用システムでは，軽自動車税システムにおいて

申告の遅れや，関係機関を経由した申告により，データの入力が遅れる場合があった．さらに，定期的なバッチ処理である修更正処理では，オンライン入力中にデータの訂正を行い，バッチ検索処理を再実行して訂正の反映された検索結果を得る必要があった．

この修更生処理に対して訂正検索を適用した結果，オンライン入力中も，以下のようにデータ訂正を反映したバッチ検索処理が可能になるという効果が確認できた．

- **異動リスト作成処理** 前回締め切り時刻以降，今回締め切り時刻までの時間区間の入力データを対象として，データ訂正を反映したバッチ検索処理の再実行が可能になった．
- **統計処理** 前回締め切り時刻時点および今回締め切り時刻時点における，データ訂正を反映したバッチ検索処理の再実行が可能になった．
- **修更正処理** 前回締め切り時刻時点での訂正を反映しない検索結果と，今回締め切り時刻時点でのデータ訂正を反映したバッチ検索処理の再実行が可能になった．

上記の例では，異動リスト作成処理は時間区間の，また統計処理は指定時刻時点の検索であり，また修更正処理では前回に業務として実施された結果と今回の結果を比較する検索になっている．ここで，修更正処理では前回の検索結果については既に住民に通知済みのため訂正は反映せず．今回の修更正については訂正を反映した．前回の訂正を反映しない再検索は，有効時間を基準日，トランザクション時間を前回バッチ処理時刻としたバイテンポラルデータベースの検索になった．このように，実際のシステム運用ではさまざまな条件での検索が必要になったが，訂正検索とバイテンポラルデータベースの検索処理を併用することで，これらの条件を指定した検索が可能になった．なお，訂正検索とバイテンポラルデータベースの検索機能で業務上必要となる検索が実行できたため，4.2.3 節のデータの直接訂正は，実際の業務運用において行う必要はなかった．

また，バイテンポラルデータベースでは業務による異動と，入力誤りの訂正は分けて管理される．訂正検索においても，図 4.19 に示すように，締め切り時刻までにオンライン入力されたデータを対象として，締め切り時刻以降の訂正のみを反映した検索を行うことができた．すなわち，オンライン入力中にも，データの訂正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．この効果として，バイテンポラルデータベースで夜間バッチとして行われていた，オンライン入力の遅れを伴う業務のバッチ処理を，締め切り時刻以降の業務時間内に実行できた．この結果，バイテンポラルデータベースを適用した運用から，さらに夜間のシステム運用負荷を削減できた．

表 4.1: 訂正検索の適用割合

No	業務区分	テーブル数	T_a	T_d
(1)	住民情報系システム	36	32(89%)	18(50%)
(2)	税関連システム	72	58(81%)	31(43%)
(3)	福祉系システム	40	37(93%)	24(60%)
(4)	内部情報系システム	63	42(67%)	8(13%)
	合計	211	169(80%)	81(38%)

4.5.2 基幹系システムに対する適用範囲の評価

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 4.1 に示す．トランザクション時間による履歴管理を行うテーブルが追加時刻 T_a を付加したテーブルであり，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 4.1 の T_d が訂正検索の適用割合になる．ここで，表の「No」は 3.3.1 節の業務区分に対応している．また，マスタテーブルのみを対象とし，以下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テーブル
- 一時的なデータを保存したテーブル
- 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%が対象であり，訂正検索の適用割合は 38%で履歴管理を行うテーブル，すなわち追加時刻を持つテーブルの約 50%が対象になった．ここで，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，(4) の内部情報系システムでは 13%に留まった．内部情報系システム以外では，軽自動車税修更生業務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，新たな訂正伝票発行による精算で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，データ訂正を分けて管理する必要がある業務で有効だった．

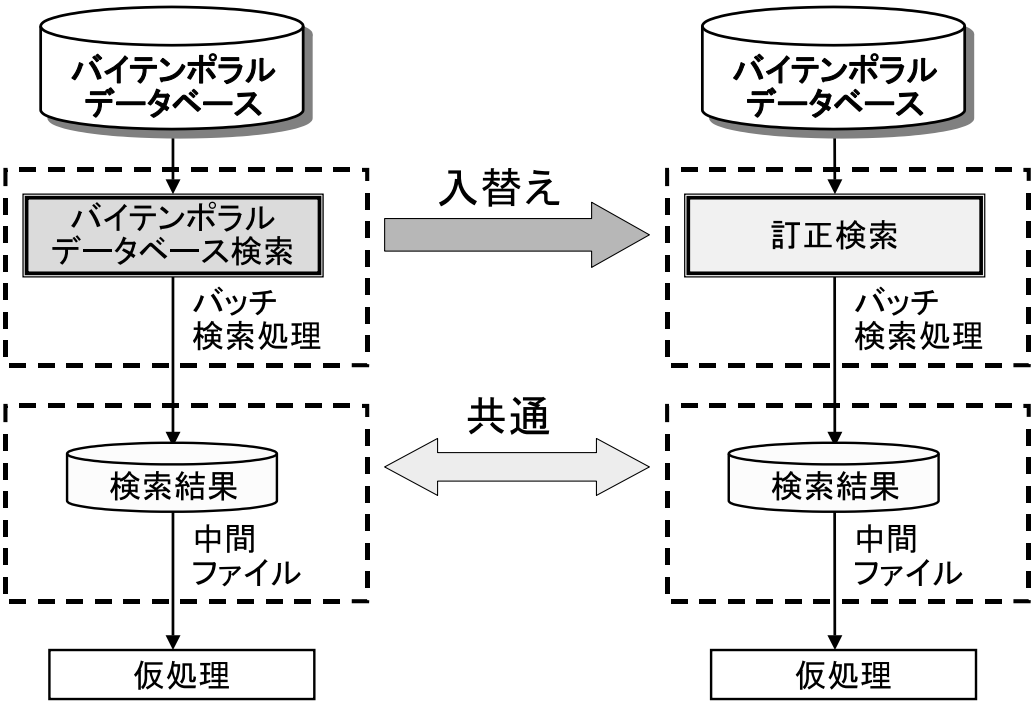


図 4.21: バイテンポラルデータベース検索の訂正検索への入替え

4.5.3 訂正検索の実装に関する評価

業務システムでは，誤ったデータを削除し，後日，スナップショット上で同じ主キー属性値を持つ，別のデータを登録することがある．従って，訂正検索では，訂正検索時刻時点と検索時刻時点のデータが，同一データか別データか識別するために，訂正前後のデータが連続していることを検証することが必要になった．これについては，図 4.12 のトランザクション時間の表現により，検索処理を容易にできた．この結果，実際の基幹系システムにおいても，商用リレーショナルデータベースの機能範囲で，訂正検索を実装することができた．

また，訂正検索はバイテンポラルデータベースにおける改良であり，バッチ処理としては図 4.15 に示すようにバッチ検索処理のジョブ追加により実装できた．従って，データベースの構成や，オンライン入力機能はそのまま流用できるため，実際の基幹系システムにおいても容易に適用が可能だった．例えば，バッチ処理では，バッチ検索処理の結果は中間ファイルに保存され，以降の仮処理や確定処理に渡される．従って，図 4.21 に示すように，中間ファイルの形式を変更せずにバイテンポラルデータベースの検索機能による

ジョブを，訂正検索処理によるジョブに入れ替えることで，訂正検索機能を容易に利用できた．

4.6 考察

バイテンポラルデータベースでは，入力の遅れるデータについてバッチ検索処理の要件を満たすことができない場合があるという課題があった．訂正検索により，この課題が解決でき，また，バッチ検索処理の要件を満たすことができることが分かった．さらに，実際の基幹系システムに適用し，実際のシステム運用において，このようなデータに対しても夜間バッチを削減できる効果を確認した．また，訂正検索は実世界の異動と，データ訂正を分けて管理する必要がある業務，すなわち，外部の申請や請求に基づいて業務を行い，結果を再び申請者や請求者に送付するような業務システムで有効であることを確認した．

外部の申請や請求に基づく基幹系システムとしては，近年，インターネット技術の進展により電子申請，電子商取引などの Web システムが急速に普及している．このようなシステムは，通常ノンストップで運用されるためオンライン入力は停止できず，また業務システムの面では従来のシステムと同様に，決算や，請求処理など，定期的なバッチ処理が発生する．今後，システムの普及，進展に伴い，このような分野でもシステム運用形態が多様化していくことが予想される．例えば，軽自動車税システムのように関係機関を経由する場合などでは，データがシステムに受付けられるまでに時間を要する．さらに，関係機関のような他システムからデータを受領する際には，1 件毎の入力だけでなくバッチ処理による一括入力もまた一般的に行われる．例えば，費用の請求や支払は定期的に行われるため，金融機関での口座引き落しは，定期的に請求機関から金融機関に一括してデータが渡され，一括処理されるという運用が一般的に行われている．このような，従来のバイテンポラルデータベースの検索だけでは対応できない運用に対しても，訂正検索を適用することにより，システムを停止することなくバッチ処理を実行できる．従って，訂正検索は今後，進展していく Web システムで有効であると考えられる．

また，バイテンポラルデータベースで構築されたシステムでは，訂正検索はバッチ検索処理におけるジョブの追加として実装できた．従って，実装にあたってはバイテンポラルデータベースのマスタを変更する必要はなく，必要に応じてバッチ検索処理のプログラムを訂正検索を利用したプログラム置き換えることで容易に実装が可能であることが分かっ

た．すなわち，訂正検索をバイテンポラルデータベースの補助機能として実装することで，バイテンポラルデータベースの検索と訂正検索の両方の検索機能が利用できることが分かった．実際の業務システムに適用したところ，4.4 節に示すように，さまざまな検索により多面的にデータの処理を行うことが必要だった．この結果から，補助機能として実装可能な訂正検索は有効であることが確認できた．

さらに，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，オンライン入力されたデータを定期的に処理する業務のように，変更，削除の訂正のみを反映したバッチ検索処理を行う場合にはトランザクション時間データベースにも適用可能である．ただし，追加による訂正を扱う場合には，データの訂正内容により属性の追加が必要になる．また，データ訂正の際に，キー情報の変更を伴う訂正が発生する場合には訂正前後のデータを対応付ける識別子が必要であり，オンライン入力漏れを訂正により追加する場合には本来の入力時間である訂正追加時間が必要になる．すなわち，トランザクション時間データベースに対する属性の追加が必要になるという課題があるが，業務データとしてこのような代替となるデータを扱う場合には，効率的に適用可能であると考えられる．例えば，訂正追加時刻は適用システムにおけるユーザ定義時刻である届出日が該当する．また，識別子としては，例えば，自動的に付番できる初回登録の端末番号とトランザクション時刻の組み合わせなどが考えられる．

4.7　むすび

システムの効率的な運用には，オンライン入力とバッチ検索処理を並行して実行できることが必要であり，第 3 章でバイテンポラルデータベースがこのための要件を満たすこと，実際の業務システムでも有効であることを示した．しかし，実際のシステム運用ではデータがオンライン入力されるまでに時間を要する場合がある．バイテンポラルデータベースでは，有効時間を使用して該当データを検索するため，指定された期日，あるいは周期でバッチ処理を行う場合には，入力の遅れたデータがバッチ処理から漏れてしまうという課題があった．

本章では，定期的なバッチ処理が締め切り時刻までに入力されたデータを対象とすることを利用し，締め切り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．これにより，バイテンポラルデータベースの課題を解決できることを示した．また，訂正検索はバイテンポラルデータベースの改良であり，バッチ検索処理に訂正検索

のジョブを追加ことで，バイテンポラルデータベースの検索機能と併用できることを示した．さらに，基幹系システムに適用し，実際の業務運用においても業務時間外のバッチ処理処理を削減できる効果があることを確認した．

これにより，バイテンポラルデータベースでは夜間バッチとして処理されていた，データの入力遅れのある業務のバッチ検索処理が，他の業務と同様に業務時間内に処理できるようになり，システム運用負荷を削減する効果があることが分かった．さらに，オンライン入力だけでなく一括入力されたデータにも適用できること，実世界の異動とデータ訂正を分けて管理する必要のある業務で有効であることが分かった．

第5章　結論

情報システムの構築においては，システムの機能性向上とコスト削減の関係や，標準パッケージの採用と独自の運用の関係など，対立する要件の中で最適解を求めなければならない場合が多くある．本研究では，このような問題の１つである，基幹系システムにおいて多数のユーザへのサービスを行うオンライン入力と，大量データを一括処理するためのバッチ検索処理の問題を扱った．従来，これらの処理は方式が異なるため，バッチ検索処理はオンライン入力を伴う業務の終了後に夜間バッチとして実施されていた．しかし，近年ではWebシステムの進展や情報システムのサービス範囲の拡大により夜間にもオンライン入力を行う運用が普及し，夜間バッチの時間に関する制約が厳しくなってきている．本論文では，双方の処理に関する最適解として，両者を同時に実行できる方式の提供を目的とした．そのための手段として，バイテンポラルデータベースを用いた検索方式の評価および改良の提案を行った．本章では，これまで議論した内容をまとめるとともに，今後の課題を述べる．

5.1　本研究のまとめ

第１章では，基幹系システムの運用を検討し，多数の端末から並行してデータを入力するためのオンライン入力と，大量のデータを効率良く処理して統計資料などを作成するためのバッチ処理という異なった処理形態を採用する必要があること，オンライン入力時間の延長に対応するため，オンライン入力とバッチ検索処理を並行して実行できることが必要であることを示した．また，オンライン入力と並行してバッチ検索処理を実行するため

の従来技術である，レコード・ロック，ミニバッチ，マルチバージョン同時実行制御などを概観し，従来技術では実際のシステム運用を行う上で課題があることを示した．

第２章では，本研究で解決すべき課題を示し，本研究の基礎となっている時制データベースの研究を概観した．まず，実際のシステム運用におけるバッチ処理ではデータ誤りなどのさまざまな運用上の問題が発生するため，オンライン入力と並行してバッチ検索処理を実行するためには以下の要件を満たす必要があることを示した．

- **バッチ検索処理の要件**

  要件１ オンライン入力との並行実行

  要件２ データ訂正時の再実行

  要件３ スケジュールの柔軟性

  要件４ 実際の業務システムへ適合性

また，時制データベースは時系列に変化するデータを管理するデータベースであり，ある事実がデータベース内に存在していた時間である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，ある事実が実世界において有効だった有効を管理する有効時間データベース，トランザクション時間と有効時間の双方を管理するバイテンポラルデータベースに分類される．これらのデータベースのうち，トランザクション時間データベースは要件２と要件３に関する課題があること，有効時間データベースは従来のバッチ検索処理技術を使用した検索を行う必要があるため従来技術と同様の課題が発生することを示した．また，バイテンポラルデータベースの実装には検索の複雑化や，データ量の増大という課題があり，実際の基幹系システムに適用，評価した事例がないため要件４が検証されていないことを示した．

第３章では，バイテンポラルデータベースを基幹系システムである自治体システムに適用し，評価，考察した．その結果，商用リレーショナルデータベース上で，基幹系システムに適用可能なバイテンポラルデータベースを構築できることを確認した．ここで，実装上の課題に対しては，バッチ処理の中間ファイルをワークテーブルで構成し段階的にデータを処理して検索を単純化し，有効時間の表現を時間区間ではなくイベント型にすることで履歴の発生を抑止した．また，実際のシステム運用においてバッチ検索処理の要件を満足することを確認した．この結果，夜間バッチの削減や，柔軟なスケジュールによりシステム運用負荷を削減でき，有効であることを確認した．さらに，長期に渡るデータ訂正を伴う業務に対しては異動統計などの訂正データの把握に効果があること，データの訂正履

歴を，内部処理での訂正と業務処理での訂正の両面から管理できること，複数の時刻現在のデータを統合した処理が実行できること，というデータ管理上の効果があることを確認した．

一方，バイテンポラルデータベースによるバッチ検索処理では，有効時間によって対象データを抽出するため，バッチ検索処理実行までに実世界のデータがシステムに入力される必要がある．しかし，適用の結果，実際のシステム運用ではオンライン入力までに時間を要する業務があり，バイテンポラルデータベースによるバッチ検索処理を適用できない場合があることが分かった．適用システムでは有効時間の代わりにユーザ定義時間である自治体への届出日を使用する方式により，入力に時間を要するデータもバッチ検索処理の対象にした．ただし，業務によっては届出からオンライン入力までに時間を要するものや，届出日のような有効時間を代替できる属性がないものがあった．これらの業務について，定期的なバッチ処理を行う場合には，処理対象から漏れるデータが発生するという課題があることが分かった．

第４章では，バイテンポラルデータベースの課題の改良として，訂正検索を提案した．これは，定期的なバッチ処理が締め切り時刻までに入力されたデータを対象とすることを利用し，締め切り時刻現在の検索結果に，それ以降に発生した訂正を反映する方式である．これにより，データの入力遅れのある業務のバッチ検索処理も，他の業務と同様に業務時間内に処理できるようになり，システム運用負荷を削減する効果があることが分かった．また，訂正検索は，バイテンポラルデータベースの検索機能に付加することで実装でき，バイテンポラルデータベースの検索と訂正検索の双方の検索方式を同時に利用することができることを示した．なお，訂正検索は一旦，入力したデータを物理的に訂正することなく，検索の際に必要な訂正を反映する方式である．従って，従来のバイテンンポラルデータベースや，トランザクション時間データベースの持つ，データの訂正経緯の把握や，データベースの監査性などの基幹系システムで必要な特性は維持される．

以上のように，本論文では基幹系システムにおいてオンライン入力とバッチ検索処理を並行して実行するための方式を議論した．本論文で議論した方式は，第１章に示したスナップショットデータベースにおける，レコード・ロック，ミニバッチ，マルチバージョン同時実行制御，第２章に示した有効時間データベースと，トランザクション時間データベースのスナップショット，第３章に示したバイテンポラルデータベースのスナップショット，第４章に示したバイテンポラルデータベースにおける訂正検索がある．これらの方式

データベースの区分		検索方式	4 実際の業務システムへの適合性	1 オンライン入力と同時実行	3 スケジュールの柔軟性	2 データ訂正時の再実行	ー 入力遅れのあるデータへの対応
スナップショットデータベース		レコード・ロック					
		ミニバッチ					
		マルチバージョン同時実行制御	●	●			
時制データベース	有効時間データベース	スナップショット＋レコード・ロック					
		スナップショット＋ミニバッチ					
		マルチバージョン同時実行制御	●	●			
	トランザクション時間データベース	スナップショット	●	●	●		
	バイテンポラルデータベース	**スナップショット**	●	●	●	●	
		スナップショット ＋ 訂正検索	●	●	●	●	●

図 5.1: データベース・検索方式とバッチ検索処理要件の対応

のバッチ検索処理の要件との対応を図 5.1 に示す．図で「●」は要件を満たすことができることを示す．

まず，スナップショットデータベースについては，リレーショナルデータベースのトランザクション機能による，レコード・ロック，あるいはバッチ検索処理を短時間のトランザクションに分割して実行するミニバッチがある．しかし，これらの方式では，オンライン入力中にバッチ検索処理中を行う場合には検索中に一部のデータが更新され，一貫性のある検索結果が得られないという問題が発生する．一方，マルチバージョン同時実行制御を使用することにより，オンライン入力中にも検索開始時点の一貫性のある検索を行うことができる．すなわち，要件 1 のオンライン入力との同時実行を満たす．ただし，検索すべき時刻を厳密に指定したり，過去の指定時刻現在の状態を検索したりすることはできない．

次に，時制データベースのうち，有効時間データベースではデータベース内におけるデータの履歴を管理していないため，バッチ検索処理ではスナップショットデータベースと同様になる．一方，トランザクション時間データベースでは検索すべき時刻を厳密に指

定したり，過去の指定時刻現在の状態を検索したりすることが可能であり，要件 3 のスケジュールの柔軟性を満たす．しかし，要件 2 のデータ訂正時の再実行のためには，バイテンポイラルデータベースによる検索方式が必要になる．ただし，バイテンポラルデータベースの検索方式では，実世界の状態をバッチ処理を開始するまでにシステムに入力する必要がある．従って，入力の遅れるデータがある場合には，訂正検索が必要になる．

ここで，各々のデータベースおよび検索方式は，実装に要するコストが異なる．例えば，レコード・ロック，マルチバージョン同時実行制御はリレーショナルデータベースの DBMS に広く実装されている．一方，トランザクション時間データベースを使用する場合にはトランザクション時間をユーザに非公開にする必要があり，専用の DBMS を使用するか DBMS の実装が必要になる．さらに，バイテンポラルデータベース，あるいは訂正検索を利用する場合には，第 3 章あるいは第 4 章に示した実装が必要になる．ここで，実際のシステムに適用する場合には，業務の運用形態により必要となるバッチ検索処理の要件が異なる．すなわち，適用システムの要件に応じて，適切なバッチ検索処理の方式を選択することが必要である．

また，オンライン入力とバッチ検索処理の今後の運用を展望するとき，インターネットによる基幹系システムの進展やユビキタス・コンピューティングの流れのなかで，ユーザがいつでも，どこでも基幹系システムを利用するという方向への流れが加速していくと考えられる．すなわち，基幹系システムは不特定多数のユーザからのオンライン入力をノンストップで受付けることになり，一方で基幹系システムの適用範囲の広がりによりバッチ検索処理ではデータ量の増大や，システム運用の多様化が進んでいくことが予想される．このような流れの中では，さまざまな利用環境で両者の処理を並行して実行するための方式が重要になる．従って，本研究の成果に基づく検索方式や実装は，今後，Web システムによる基幹系システムを構築する上で有効になると考えられる．

5.2 今後の課題

本論文ではオンライン入力中のバッチ処理の中で，データベースの検索を行う処理を扱ったが，実際のシステム運用ではバッチ処理によるデータベースの更新もまた行われる．例えば，4.4 節の図 4.18 に示した適用システムの軽自動車税の事例では，関係機関からの車両異動データを車両テーブルに入力する際には一括入力を併用したため，バッチ処

理によるデータベースの更新（以下，バッチ更新処理と略記）が行われた．この結果，車両テーブルはオンライン入力とバッチ更新処理の両方から更新されたため，バッチ更新処理はオンライン入力を伴う業務終了後に実施した．

バッチ更新処理においても，データ１件毎にトランザクションとして更新する方式や，1.2 節に示したミニバッチにより更新する方式が提案されている．しかし，これらの方式では，相互に関係するデータ，例えば軽自動車１台についての複数の履歴を１つのトランザクションとして更新できないため，更新中の状態が他の処理から参照されＡＣＩＤ特性のうちの隔離性が維持できないという課題がある．さらに，障害発生時にはロールバックを行う必要があり，サーガによる補償トランザクションの方式が提案されている．しかし，この方式では更新済みデータを参照した処理が行われるため完全な復元ができないという課題がある [70]．

また，本論文で提案した訂正検索は，トランザクション時間データベースにも適用可能である．ただし，トランザクション時間データベースにおいて，追加の訂正を行う場合には属性の追加が必要になるという課題が残った．さらに，本論文では時制データに関する訂正の反映の観点から議論したが，各適用事例で示したとおり時制データについては訂正の反映だけでなく，さまざまな観点からの検索が必要である．例えば，訂正検索はあるトランザクション時間のスナップショットに対する訂正の反映であるが，軽自動車税の修更正に関するデータ管理では指定された有効時間現在に関する訂正の履歴の検索が行われる．このような，さまざまな時制データに関する検索表現の一般化が今後の課題として残されている．

すなわち，今後の課題としては，まず，全てのバッチ処理をオンライン入力と並行して実行するため，オンライン入力と並行実行可能なバッチ更新処理方式の研究がある．次に，トランザクション時間データベースへの効率的な訂正検索の実装方式の研究と，時制データに関する一般化した検索表現の研究がある．

付 録A　時制データベースに関する用語

時制データベースは，当初，用語や概念が充分に統一されないまま各方面での研究が進められた．このため，1992 年から関係者により概念と用語の統一を図るための努力が行われ，1998 年に「The Consensus Glossary of Temporal Database Concept – February 1998 Version」[19] がまとめられた．しかし，それ以前の論文では，さまざまな用語が使用されたため，同一の用語が文献によって異なった概念を表している場合がある．本論文では，上記の文献に基づき，下記の「–」の用語を利用する．なお，本論文の用語に併せて,「*」に同一概念の他の用語を記載する [19, 33].

- 本論文で使用する用語
  - 有効時間（valid time）
    - 実時間, real-world time, intrinsic time, logical time, data time
  - トランザクション時間（transaction time）
    - registration time, extrinsic time, physical time, transaction commit time, transaction–time
  - ユーザ定義時間（user–defined time）
    - なし
  - スナップショットデータベース（snapshot database）
    - database, conventional database, static database
  - 時制データベース（temporal database）

    - time–oriented database, historical database
- 有効時間データベース（valid time database）
    - historical database，valid–time database
- トランザクション時間データベース（transaction time database）
    - rollback database, transaction–time database
- バイテンポラルデータベース（bitemporal database）
    - temporal database，fully temporal database，valid–time and transaction–time database，valid–time transaction–time database

謝辞

本研究においてご指導ご鞭撻を賜りました静岡大学情報学部　水野忠則博士に深く感謝申しあげます．本研究の過程において終始適切なご助言とご鞭撻を賜り，また，研究をまとめるに当たり懇切なるご指導と励ましを賜りました，東海大学情報理工学部　片岡信弘博士に深く感謝申し上げます．本論文をまとめるにあたり，懇切なるご指導を賜りました静岡大学情報学部　酒井三四郎博士に深く感謝申しあげます．また，本研究の国際会議での発表と，本論文をまとめるにあたり，ご支援いただきました静岡大学情報学部　峰野博史博士，三菱電機インフォメーションテクノロジー株式会社　石野正彦博士に深く感謝申し上げます．

静岡大学大学院博士課程への社会人入学にあたり，入学をご許可，ご支援いただきました三菱電機インフォメーションシステムズ株式会社（当時）　笠井鯉太郎博士，堂坂辰氏，木梨隆司氏，三菱電機インフォメーションシステムズ株式会社　阿部恵成氏に深く感謝申しあげます．本研究は，三菱電機株式インフォメーションシステムズ株式会社の各位のご支援のもとで行わせていただきました．特に，国際会議での発表を許可頂き，ご指導と励ましをいただきました下間芳樹博士，高野茂樹氏，村田幸久氏，不在の間，業務を代行いただきました見戸義英氏，桜庭伸一郎氏，本間敏夫氏に深く感謝申し上げます．

また，本研究のアイデアは三菱電機株式会社の自治体システム開発を担当した際に，夜間バッチ処理をなくし楽に運用できるシステムを提供するという構想から生まれたものです．多くの議論を通じてアイデアの具体化を支援いただいた開発関係者各位に深く感謝申し上げます．各位の熱意によって実用に供するシステムとすることができ，各地の自治体の実際の運用を通じてバイテンポラルデータベースに関するさまざまな知見を得ることができました．特に，本システムの開発の機会を与えていただき，ご指導ご鞭撻を賜りました三菱電機株式会社（当時）　矢島武氏に深く感謝申し上げます．また，システム開発にあたり中心となって推進いただきました三菱電機株式会社　濱崎光幸氏，秋間孝道氏，鈴木篤氏，白川和重氏，三菱電機インフォメーションシステムズ株式会社　三浦健二氏，芝田哲也氏，山口浩氏，井料知己氏，久元武司氏に深く感謝申し上げます．また，アイデ

アのポイントを整理する過程で協力いただき，その効果と特徴について貴重なご意見をいただきました三菱電機株式会社　川田俊尚氏に深く感謝申し上げます.

最後に，本研究を進めるにあたり協力いただいた妻　尚代と，麻莉恵，隆史，そして励まし続けていただいた父　克己と，亡き母　愛子に深く感謝いたします.

参考文献

[1] Agrawal, D., Mark, L. and Roussopoulos, N.: Incremental Implementation Model for Relational Database with Transaction Time, *IEEE Trans. knowledge and Data Eng.*, Vol. 3, No. 4, pp. 461–473 (1991).

[2] Van den Akker, J. and Siebes, A.: Object histories as a foundation for an active OODB, *Proc. 7th International workshop on Database and Expert Systems Applications*, pp. 2–8 (Sep. 1996).

[3] Bækgaard, L. and Mark, L.: Incremental Computation of Time–Varying Query Expressions, *IEEE Trans. knowledge and Data Eng.*, Vol. 7, No. 4, pp. 583–590 (1995).

[4] Bettuni, C. and et. al.: Summaries of Current Work–The Dagstuhl Semminar Reasearchers, *Temporal Database: Research and Practice.(the book grow out of a Dagstuhl Seminar, June 23–27, 1997).* Lecture Notes in Computer Science 1399, Springer–Verlag, pp. 414–428 (1998)

[5] Bhargava, G. and Gadia, S. K.: Relational Database Systems with Zero Information Loss, *IEEE Trans. knowledge and Data Eng.*, Vol. 5, No. 1, pp. 76–87 (1993).

[6] Böhlen, M. and Marti, R.: On the Completeness of Temporal Database Query Languages, *Tempral Logic: First International Conference, ICTL '94* Lecture Notes in Artificial Intelligence 827, Springer–Verlag, pp. 283–300 (July 1994)

[7] Chittaro, L. and Combi, C.: Temporal Indeterminacy in Deductive Database: An Approach Based on Event Calculus, *Active, Real–Time, and Temporal Database Systems: Second International Workshop, ARTDB–97* Lecture Notes in Computer Science 1553, Springer–Verlag, pp. 212–227 (Sep. 1997)

[8] Chiueh, T. and Pilania, D.: Design, Implementation, and Evaluation of a Repairable Database Management System, *Proc. 21st International Conference on Data Engineering*, pp. 1024–1035 (2005)

[9] Chomicki, J.: Temporal Query Languages: a Survey, *Tempral Logic: First International Conference, ICTL '94* Lecture Notes in Artificial Intelligence 827, Springer–Verlag, pp. 506–534 (July 1994)

[10] Chomicki, J. and Toman, D.: Implementing Temporal Integrity Constraints Using an Active DBMS, *IEEE Trans. knowledge and Data Eng.*, Vol.7, No.4, pp. 566–581 (Aug. 1995).

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

[12] Eder, J. and Koncilia, C.: Representing Temporal Data in Non–Temporal OLAP Systems, *Proc. 13th International Workshop on Database and Expert Systems Applications*, pp. 817–821 (2002).

[13] Finger, M. and MçBrien, P.: Concurrency Control for Perceivedly Instantaneous Transactions in Valid–Time Databases, *Proc. 4th International Workshop on Temporal Representation and Reasoning*, pp. 112–118 (1997).

[14] Gadia, S.K. and Nair, S.S.: Algebraic Identities and Query Optimization in a Parametric Model for Relational Temporal Databases, *IEEE Trans. knowledge and Data Eng.*, Vol. 10, No. 5, pp. 793–807 (Sep./Oct. 1998).

[15] Gal, A. and Etzion, E.: A Multiagent Update Process in a Database with Temporal DAta Dependencies and Schema Version, *IEEE Trans. knowledge and Data Eng.*, Vol. 10, No. 1, pp. 21–37 (Jan./Feb. 1998).

[16] Gawne Cain Research Ltd.: Herodotus, http://herodotus.biz/Overview.html.

[17] Han, H., Park, S. and Park, C.: A concurrency control protocol for read–only transactions in real-time secure database systems, *Seventh International Conference on Real–Time Computing Systems and Applications*, pp. 458–462 (Dec. 2000).

[18] Jensen, C.S., Mark, L. and Roussopoulos, N.: Incremental Implementation Model for Relational Database with Transaction Time, *IEEE Trans. knowledge and Data Eng.*, Vol. 3, No. 4, pp. 461–473 (1991).

[19] Jensen, C.S., Dyreson, C.E. and et. al.: The Consensus Glossary of Temporal Database Concept – February 1998 Version, *Temporal Database: Research and Practice.(the book grow out of a Dagstuhl Seminar, June 23–27, 1997).* Lecture Notes in Computer Science 1399, Springer–Verlag, pp. 367–405 (1998)

[20] Jensen, C.S. and Snodgass, R.T.: Temporal Data Management, *IEEE Trans. knowledge and Data Eng.*, Vol. 11, No. 1, pp. 36–44 (1999).

[21] Johnson, R.G. and Lorentzos, N.A.: Comments on "Extensions to SQL for Historical Databases", *IEEE Trans. knowledge and Data Eng.*, Vol. 4, No. 4, pp. 220–230 (Aug. 1992).

[22] Kobayashi, H.: Safety Analysis using Temporal Database, 信学技報, Vol.101, No.132, pp. 17–23（June 2001）.

[23] Lomet, D., Barga , R., Mokbel, M.F., Shegalov, G., Wang, R. and Zhu, Y.: Transaction Time Support Inside a Database Engine, *22nd International Conference on Data Engineering*, pp. 35–46 (Apr. 2006).

[24] Luttermann, H. and Grauer, M.: Using Interactive, Temporal Visualizations for WWW–based Presentation and Exploration of Spatio–Temporal Data, *Spatio–Temporal Database Management: International Workshop STDBM'99* Lecture Notes in Computer Science 1678, Springer–Verlag, pp. 100–118 (Sep. 1999)

[25] Makni, A., Bouaziz, R. and Gargouri, F.: Formal Verification of an Optimistic Concurrency Control Algorithm using SPIN, *Proc. Thirteenth International Symposium on Temporal Representation and Reasoning*, pp. 160–167 (2006).

[26] Martín, C. and Sistac, J.: An Integrity Constraint Checking Method for Temporal Deductive Databases, *Proc. 3rd International Workshop on Temporal Representation and Reasoning*, pp. 136–141 (1996).

[27] Oracle：Oracle Database SQL リファレンス 10g リリース 2（10.2） B19201–02, http://otndnld.oracle.co.jp/document/products/oracle10g/102/doc_cd/server.102/B19201–02/title.html （Mar. 2006）.

[28] Özsoyoğlu, G. and Snodgrass, R.T.: Temporal and Real–Time Databases: A survey, *IEEE Trans. knowledge and Data Eng.*, Vol. 7, No. 4, pp. 513–532 (Aug. 1995).

[29] Park, H.-J. and Yoo, S.I.: Implementation of Checkout/Checkin Mechanism on object-Oriented Database System, *Proc. 7th International workshop on Database and Expert Systems Applications*, pp. 298–303 (Sep. 1996).

[30] Sarda, N.L.: Extensions to SQL for Historical Databases, *IEEE Trans. knowledge and Data Eng.*, Vol. 2, No. 2, pp. 220–230 (June 1990).

[31] Shrira, L. and Xu, H.: SNAP: Efficient Snapshots for Back–in–Time Execution, *Proc. 21st International Conference on Data Engineering*, pp. 434–445 (2005).

[32] Skaf, H., Charoy, F. and Godart, C.: Flexible Integrity Control of Cooperative Applications, *Proc. 9th International Workshop on Database and Expert Systems Applications*, pp. 901–906 (1998).

[33] Snodgrass, R. and Ahn, I.: Temporal Databases, *IEEE COMPUTER*, Vol. 19, No. 9, pp. 35–42 (Sep. 1986).

[34] Snodgrass, R., Gomez, S. and McKenzie, E.: Aggregates in the Temporal Query Language TQuel, *IEEE Trans. knowledge and Data Eng.*, Vol. 5, No. 5, pp. 826–842 (Oct. 1993).

[35] Snodgrass, R.T.: Developing Time–Oriented Database Applications in SQL, Morgan Kaufmann Publishers (Sep. 1999).

[36] Snodgrass, R.T., Böhlen, M.H., Jensen, C. S. and Steiner, A.: Transitioning Temporal Support in TSQL2 to SQL3, http://www.cs.aau.dk/ csj/Thesis/pdf/chapter25.pdf.

[37] Srinvasan, J., Das, S. and Freiwald, C.: Oracle8*i* Index–Organized Table and its Application to New Domains, *Proc. 26th International Conference on Very Large Database*, pp. 285–296 (2000).

[38] Stantic, B., Thornton, J. and Sattar, A.: 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*, pp. 174–180 (2003).

[39] Tansel, A.U.: Integrity Constraints in Temporal Relational Databases Extended Abstract, *Proc. Int. Conf. on Information Technology: Coding and Computing*, pp. 460–464 (2004).

[40] Terenziani, P.: Symbolic User–Defined Periodicity in Temporal Relational Databases, *IEEE Trans. Knowledge and Data Eng.*, Vol. 15, No. 2, pp. 489–509 (2003).

[41] Varman, P.J. and Verma, R.M.: An Efficient Multiversion Access Structure, *IEEE Trans. knowledge and Data Eng.*, Vol. 9, No. 3, pp. 391–409 (May/June 1997).

[42] Wang, L. and Peng, P.: Extension of Multi–Version Concurrency Control Mechanisms for Long–Duration Transaction Based on Nested Transaction Model, *Fourth International Conference on Computer and Information Technology*, pp. 963–968 (Sep. 2004).

[43] Wu, Y., Jajodia, S. and Wang, X.S.: Temporal Database Bibliography Update, *Temporal Database: Research and Practice.(the book grow out of a Dagstuhl Seminar, June 23–27, 1997).* Lecture Notes in Computer Science 1399, Springer–Verlag, pp. 338–366 (1998)

[44] Xu, H. and Furukawa, T.: Supporting Cooperative Work Based on Semantics of Workflows, *Proc. 1999 International Symposium on Database Application in Non–Traditional Environments*, pp. 366–369 (2000).

[45] Yadav, D.S., Agrawal, R., Chauhan, D.S., Saraswat, R.C. and Majumdar, A.K.: Modeling Long Duration Transactions with Time Constraints in Active Database, *Proc. International Conference on Information Technology: Coding and Computing*, Vol.1, pp. 497–501 (2004).

[46] Yang, J.-Y., Lee, I.-H., Jeong, O.-R., Song, J.-Y., Lee, C.-M. and Lee, S.-G.: An architecture for supporting batch query and online service in Very Large Database systems, *IEEE International Conference on e–Business Engineering*, pp. 549–553 (2006).

[47] 天笠俊之，田頭利規，金森吉成，増永良文：制約を導入した時区間代数，信学技報，Vol. 95, No. 148，pp. 41–48（July 1995）

[48] 天笠俊之，有次正義，田中貴之，金森吉成：時空間概念データモデルの実装，情処学論，Vol. 40, No. SIG6 (TOD3)，pp. 141–151 (Aug. 1999).

[49] 天笠俊之，有次正義，金森吉成：時間的に変化するデータに対する索引技術，情報処理，Vol. 42，No. 10，pp. 972–979 (2001).

[50] 井伊克益，北川博之：版管理機能を持つデータベースシステムにおける視覚的問い合わせ言語の設計，実装，評価，情処学論，Vol. 37，No. 1，pp. 110–122 (1996).

[51] 石川博，久保田和己，手塚正義：オブジェクト指向データベースとリレーショナルデータベースの相互利用について，情報処理学会データベース・システム研究会報告，Vol. 93，No. 65，pp. 145–154 (July 1993).

[52] 石井一夫：図解　よくわかるデータマイニング，日刊工業新聞社 (2004).

[53] 石原秀男，魚田勝臣，大曽根匡，斉藤雄志，出口博章，綿貫理明：コンピュータ概論－情報システム入門，共立出版 (2004).

[54] 一瀬益夫：新版現在情報リテラシー，同友館 (2006).

[55] 上西 康太，境 美樹，伊織 生美：「バッチ制御システムにおける適応的スケジューリング方式の検討」，情報処理学会グループウェアとネットワークサービス研究会報告，Vol. 2007, No. 2007–GN–65 (Sep. 2007)

[56] 宇田川佳久：－ポスト・リレーショナル・データベース－オブジェクト指向データベース入門，ソフト・リサーチ・センター (1992).

[57] 榎並利博：自治体のＩＴ革命，東洋経済新報社 (2000).

[58] 大場 達生：データ更新を伴う Operational OLAP，情報処理学会データベース・システム研究会報告，Vol. 99, No. 61，pp. 339–344 (July 1999).

[59] 大山敬三：情報検索システムのオンライン更新における検索集合の一貫性について，情報処理学会情報学基礎研究会報告，Vol. 97, No. 40，pp. 1–6 (May 1997).

[60] 大山敬三：情報検索システムのオンライン更新における一貫性維持方式，学術情報センター紀要，No. 10，pp. 29–35 (1998).

[61] 大和田尚孝，渡辺亨靖，小原忍：特集１　解体！レガシー・バッチ，日経コンピュータ，No. 659，pp. 32–47 (Aug. 21 2006)

[62] 奥川峻史，桜井哲真：デジタル情報学概論–２１世紀，ＩＴ社会理解のために，共立出版 (2000).

[63] 鬼塚真，磯部成二：長期トランザクションにおけるチェックイン・チェックアウトの自動化と短期トランザクションの実行順序の保障方式，情報処理学会データベース・システム研究会報告，Vol. 1997，No. 7，pp. 33–40 (Jan. 1997).

[64] 柏木恵：自治体のクレジット収納—導入・活用の手引き，学陽書房 (2002).

[65] 片岡良治，佐藤哲司，井上潮：部分更新と全数検索の混在処理に適した多版並行処理制御方式，信学論，(D–I)，Vol. J74–D–I，No. 3，pp. 224–231 (Mar. 1991).

[66] 鎌倉市：住民票の異動の手続き，
http://www.city.kamakura.kanagawa.jp/siminka/juidozen.htm.

[67] 鎌倉市：軽自動車税，http://www.city.kamakura.kanagawa.jp/siminzei/keiji.htm.

[68] 川上潤司：特集２３大巨額損失事件は防げた—不正を見抜く情報システム，日経ストラテジー，No.54，pp. 116–125（Sep. 1996）.

[69] 北川博之，田中肇，大保信夫，鈴木功：履歴データ型を用いた版管理データモデルの提案，情処学論，Vol. 34，No. 5，pp. 1031–1044 (1993).

[70] Gray, J. and Reuter, A.(著)，喜連川優（監訳）：トランザクション処理　概念と技法，日経ＢＰ (2001).

[71] 國枝和雄，畑田孝幸，大久保英嗣，津田孝夫：適応型時刻印方式に基づく同時実行制御，情処学論，Vol. 33，No. 6，pp. 802–811 (1992).

[72] 経済産業省：電子商取引及び情報財取引等に関する準則,
http://www.meti.go.jp/policy/it_policy/ec/070405zyunskusyusei.pdf（Mar. 2007）.

[73] 佐藤健:トランザクション処理の標準化動向，情報処理，Vol. 28，No. 4，pp. 491–497 (Apr. 1987).

[74] 佐藤哲司，片岡良治，平野泰宏，井上潮：メモリ共有型マルチプロセッサにおけるオンラインとバッチ処理の混在した実行制御法，情報処理学会データベース・システム研究会報告，Vol. 1992，No. 61，pp. 141–150 (July 1992).

[75] 自治体国際化協会：Local Government in Japan（日本の地方自治（英語版)）（対訳），http://www.clair.or.jp/j/forum/series/pdf/j05.pdf（2006）.

[76] 芝野耕司：データベース関連技術の標準化：SQL–92 の追加機能と SQL3，情報処理，Vol.37，No.7，pp. 616–622 (July 1996).

[77] 徐海燕，古川哲也，史一華：マルチバージョンデータベースにおけるワークフローに基づく並行処理制御，情処学論，Vol. 42，No. SIG4，pp. 27–35 (2001).

[78] 徐海燕，古川哲也：ワークフロートランザクションの隔離性，情処学論データベース，Vol. 44，No. SIG8 (TOD18)，pp. 55–64 (2003).

[79] 鈴木浩，渡邊進：ＩＴが拓く電力ビジネス革命，オーム社 (2002).

[80] 総務省：ｕ–Ｊａｐａｎ推進計画２００６，
http://www.soumu.go.jp/s–news/2006/pdf/060908_3_1.pdf （Sep. 2006）.

[81] 高橋三雄：ビジネス情報技術，日科技連出版社 （1996）.

[82] 滝沢誠：データベースシステム入門技術解説—基礎から詳細技術まで—，オーム社（1991）.

[83] 田中克己：組系列に基づく履歴データベースモデルとその完全性制約，情報処理，Vol. 26，No. 11，pp. 1273–1280 (1985).

[84] 田中聡，業績責任会計における時制マスタ管理システムの重要性，情報処理学会データベース・システム研究会報告，Vol. 99, No. 61，pp. 333–338 (July 1999).

[85] 谷口俊一郎，澤井良二，石川辰雄，鈴木広司：アプリケーション開発を成功に導くシステム基盤構築のノウハウ，日経ＢＰ社 (2005)

[86] 出口彰，大森匡，星守：ストレージシステムにおけるデータ一貫性を保障したアーカイブ取得方式，信学論，(D–I)，Vol. J88–D–I，No.3，pp. 698–714 (Mar. 2005).

[87] 電子政府利用支援センター：電子政府の推進について，
http://www.e-gov.go.jp/doc/scheme.html.

[88] 中川慶一郎，生田目崇：データベースマーケティングの進展，信学会誌，Vol. 87, No. 1，pp. 53–57 (Jan. 2004).

[89] 中川路哲男:クライアント/サーバ型の分散トランザクション処理，情報処理，Vol.36, No.7，pp. 633–643 (July 1995).

[90] 中里秀則:実時間データベースの並行処理制御における優先度割付け，信学論，(D–I)，Vol. J78–D–I，No.8，pp. 670–678 (Aug. 1995).

[91] 長須賀弘文，吉澤康文，新井利明，今井和男：入出力の仮想化と並列処理によるバッチ処理の高速化機能：ＰＲＥＳＥＴ，情処学論，Vol. 35, No. 5，pp. 856–864 (May 1994).

[92] 中村泰明，出木原裕順：時空属性を持った空間データの管理構造　－ＰＭＤ木－，情処学論，Vol. 40, No. SIG5 (TOD2)，pp. 54–68 (May 1999).

[93] 中村仁之輔，芳西崇，梅本佳宏，小林信幸，佐藤文明，水野忠則：ネットワークサービス向けメモリ常駐型リレーショナルＤＢＭＳの設計と実装，情処学論，Vol. 43, No.SIG5 (TOD14)，pp. 134–144 (June 2002).

[94] 西川洋一：図解・標準－最新データベース技術，秀和システム (2002).

[95] 林紘一郎，湯川抗，田川義博：進化するネットワーキング－情報経済の理論と展開，NTT 出版 (2006).

[96] 春本要，八幡孝，西尾章治郎：協調作業支援のためのデータ管理モデル，信学論，(D–I)，Vol. J79–D–I，No. 5，pp. 271–279 (1996).

[97] Bacon, J.（著），藤田昭平，篠田陽一，今泉隆史（訳）：並行分散システム－オペレーティングシステム，データベース－分散マルチメディアシステムへの統合アプローチ，トッパン（1996）.

[98] 穂鷹良介：データベース関連技術の標準化：データベース関連技術の標準化の概要，情報処理，Vol. 37，No. 7，pp. 605–615 (July 1996).

[99] 前川守:岩波講座 ソフトウェア科学7－ソフトウェア実行/開発環境，岩波書店 (1992).

[100] 増永良文:リレーショナルデータベースの基礎－データモデル編－，オーム社 (1990).

[101] 増永良文：マルチメディアデータベースと時間，情報処理，Vol. 36，No. 5，pp. 369–376 (1995）.

[102] 増永良文：Computer Science Library–14 データベース入門，サイエンス社 (2006).

[103] 丸山一夫：インターネットの光と影—被害者・加害者にならないための情報論理入門—，北大路書房 (2000).

[104] 三菱電機インフォメーションテクノロジー:ジョブ自動スケジューラ AUTORUNNER, http://www.mdit.co.jp/opencenter/autorun.htm.

[105] 三菱電機インフォメーションテクノロジー：帳票出力支援機能 FormRunner, http://www.mdit.co.jp/opencenter/formrun.htm.

[106] 三菱電機：三菱電機自治体総合行政情報システム（MTAIS–eLG）提供開始のお知らせ，http://www.mitsubishielectric.co.jp/news/2006/0328–a.htm.

[107] 三菱電機：三菱電機が考える電子自治体システム，http://www.mitsubishielectric.co.jp/jichitai/arikata/index.html.

[108] 吉沢直美，小野美由紀，石川博：オブジェクト指向データベースにおけるバージョン管理モデルの設計と実装，情処学論，Vol. 37，No. 4，pp. 556–567 (1996).

著者発表論文

＜学術論文＞

[1] 工藤司，片岡信弘，水野忠則：データ更新中の検索結果一貫性維持実現事例，情処学論データベース，Vol. 48, No. SIG11 (TOD34)，pp. 215–223 (June 2007).

＜国際会議論文＞

[1] Kudoh, T., Hirayama, M. and Mikami, K.: Evaluation of Pipelined Common Processor in Distributed Processing System, *IFAC Real Time Programming 1982*, pp. 121–127 (1982).

[2] Kudou, T., Kataoka, N. and Mizuno, T.: Integrity Maintenance System of Database Query Under Updating, *Knowledge–Based Intelligent Information and Engineering Systems: 11th International Conference, KES 2007*, Lecture Notes in Artificial Intelligence 4694, Springer–Verlag, pp. 491–498 (Sep. 2007).

[3] Kudou, T., Kataoka, N. and Mizuno, T.: Implementation of Integrity Maintenance Method of Query Result by Bitemporal Database, *Proc. International Workshop on Infomatics (IWIN2007)*, pp. 2–10 (2007).

[4] Kudou, T. and Tsukuda, Y.: Serious Failure Prevention Method of Information Systems by Analyzing Incidents, *The 4th International Conference on Project Management (ProMAC2008)*, (Sep. 2008)（採録決定）.

＜国内ワークショップ・研究会等＞

[1] 平山正治，工藤司，房岡璋：ハードウェアによるコンパイラの実現：設計とシミュレーション評価，情報処理学会計算機アーキテクチャ研究会報告，Vol. 1981，No. 17，pp. 1–10 (1981).

[2] 立花康夫，工藤司：ＦＩＲ型ピーキング･フィルタの高速処理，信学技報，CAS87–168，pp. 13–18（1987).

[3] 立花康夫，工藤司：ハニング窓の拡張と高速フィルタ・バンクへの応用，信学技報，Vol. 89，No. 10，pp. 29–36 (1989).

[4] 工藤司，片岡信弘：訂正情報を持つ履歴データベースの提案，情処研報，Vol. 2004，No. 116，pp. 17–24 (Nov. 2004).

[5] 工藤司，片岡信弘:訂正情報を付加した時制データベースの提案，信学技報，Vol. 104，No. 665，pp. 19–24 (2005).

＜全国大会等＞

[1] 川口至商，山ノ井高洋，工藤司，新保勝：錯視現象の Minkowski 幾何学的解釈，昭和 54 年度電子通信学会情報・システム部門全国大会，pp. 173 (1979).

[2] 工藤司，平山正治：並列処理計算機におけるパイプライン化した共有装置の性能評価，電子通信学会総合全国大会，pp. 227–228 (1981).

[3] 工藤司：２重インデックス向メモリ・アーキテクチャ，情報処理学会第 27 回（昭和 58 年後期）全国大会，pp. 205–206 (1983).

[4] 瀬尾和男，工藤司，平山正治：ＩＳＰＳ記述によるＰＬＡジェネレータの開発，情報処理学会第 28 回（昭和 59 年前期）全国大会，pp. 1419–1420 (1984).

[5] 工藤司，藤田博：Ｔｅｘ出力システムの開発，情報処理学会第 28 回（昭和 59 年前期）全国大会，pp. 243–244 (1984).

[6] 工藤司：セルラ・アレイ・プロセッサＣＡＰにおける大規模行列演算アルゴリズム，情報処理学会第 29 回（昭和 59 年後期）全国大会，pp. 103–104 (1984).

[7] 工藤司：セルラ・アレイ・プロセッサ用並列処理記述言語の提案，情報処理学会第 30 回（昭和 60 年前期）全国大会，pp. 469–470 (1985).

[8] 工藤司，片岡信弘，水野忠則：トランザクション時間データベースを活用したスケジュール短縮手法の提案，プロジェクトマネジメント学会 2007 年度秋季研究発表大会予稿集，pp. 156–161 (Sep. 2007).

＜表彰＞

[1] Best Paper Award, Internatioal Workshpo on Infomatics (IWIN2007) (Oct. 3 2007).

＜成立済特許＞

[1] メモリ装置，特許 1454359 (Aug. 1988).

[2] メモリアクセス制御装置，特許 1467122 (Nov. 1988).

[3] 時情報変換方式，特許 3092425 (July 2000).

[4] プログラム起動方式，特許 3386300 (Jan. 2003).

[5] データ管理システム，特許 3824468 (July 2006).