履歴更新方式を適用した企業情報システム安定化アルゴリズム

Algorithms for Stable Enterprise Information System developed by Historical Data Recording Method

堀田正隆
Masataka Hotta
株式会社プロ・アクシス
ProAxis Ltd.

要旨

先ず、企業情報システムにおいて不安定とはどういうことか。①同じ操作をしても結果が異なることがある。状況は刻々変わっていくので原因は分からないままになる②過去データは過去に取り出したものと現在取り出したものと同じでない③利用者から見て原因不明の停止が起こる（デッドロックが代表的な事例）。以上３つの不安定問題は履歴記録方式を採用することで完全に解決可能となる。このためには、リレーショナルデータベースの考え方は変えずにデータベースの設計理論だけ変える必要がある。この方式では、排他制御せずにデータの整合性を保つことが可能であり、排他制御しないことからデッドロックは起きない。情報システムの可用性を高めるとともに利用者自身による運用（ノンプロフェッショナル運用）が可能となる。

1．事実の発生と事実の記録の時間的関係

図１は企業の情報システムにおける事実の発生とその記録の時間的関係を表している。時間的関係にはＡ.定時記録Ｂ.リアルタイム記録Ｃ.不定時記録の３パターンがある。Ａ.定時記録は事実を記録する時刻が一定時に定まっており、それ以前に発生した事実は全てこの時刻までに記録すればよい。会計処理業務などはこの方式である。Ｂ.リアルタイム記録は事実の記録が即事実の発生と見做せるので事実の発生と記録の間に時間遅れ（Time Delay）が生じない。ＡＴＭ、ＰＯＳや座席予約などはこの方式である。Ｃ.不定時記録は、事実の記録が定刻でない上、事実の発生と記録の間で必ず時間遅れが生じる。遅れ幅は実務者の判断に任されていて、急ぎの時は短く（時には追い越す）するようにしているが、必ずしも次の行動に移る時までに必要な事実の記録が間に合っているとは限らず（例えば生産実績の登録が出荷の後になる）システムの安定化が難しい。生産管理などの実態はこうである。

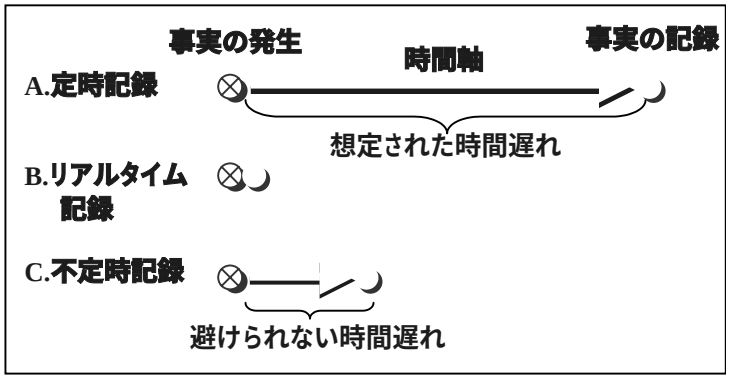

図１　事実の発生と事実の記録の時間的関係

2．システム不安定の現象

2.1．同じ操作をしても結果が異なる

システムの外からは見えないところで何らかのデータが変わっていることにより同じ操作をしても結果が異なることが起きる。図１のＣの場合は不定時にデータが変化するので防げない。替わって何のデータがどう変わったため結果が異なったかの原因追求を可能にする必要がある。このためには、前回操作時と同一の状態を再現可能にする必要がある。例えば、現在の在庫を知ることができるだけでは不十分で１時間前の在庫も知ることができる仕組みでなければならない。

1.2．過去データは過去と現在で異なる

例えば、最近５年間の売上推移表を作るとする。ここに現われる５年前の売上額は５年前に作成した売上表の金額と必ずと言って良いほど合わない。こうした不一致は製品単価や得意先別値引率などのマスターデータの不用意な上書き更新によって引き起こされる。

1.3．原因不明の停止が起こる

ハードウェアの故障は利用者にとって原因不明に当らない。プログラムのバグは論外とすると、利用者から見て原因不明と感ずる停止の多くがデッドロックに起因している。デッドロックは自動的に解除することが可能であるが、入れたデータが入ってないなどの不信感は残るので本質的な解決策とは言えない。

3．システム安定化の大前提： With Single Hand と One Fact One Place

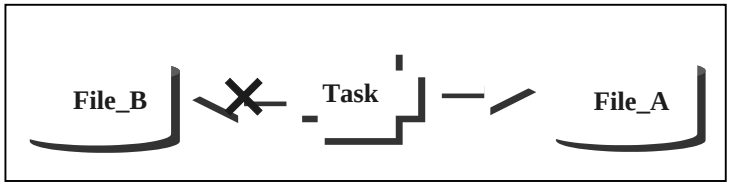

図２ With Single Hand

一つのタスク（データベースの書き換えをする利用者の操作）は唯一のファイルのみを書き換える。このことを本論ではWith Single Hand（図２）と言う。さらに With Single Hand が厳格に行われれば、One Fact One Place（一つの事実は一つのところにしかない）は自動的に成り立つ。一つのタスクは唯一のファイルのみ書き換える、がシステム安定化の大前提である。 一方、With Single Hand でない、従ってOne Fact One Placeでもない事例を次に挙げる。

ある原料が入庫されて来た。この事実は入庫実績ファイルに記録される。これによりある原料の在庫は増えるから在庫ファイルを書き換える。

入庫実績ファイルと在庫ファイルの同時更新はこれまでの情報システムの常識とされてきたことである。しかし、これは明らかにWith Single Handに違反している。さらにOne Fact One Placeにも違反している。その理由を次に述べる。在庫データは入出庫の事実を拾って加減算を行えば計算可能である。他に在庫ファイルからも取得可能である。これは厳密には One Fact **Two** Placeに他ならない。上の例の場合、在庫ファイルを更新したことが間違っている。これが無ければ排他制御をする必要もないのである。

さらに、もし入出庫ファイルが上書き更新でなく履歴更新(後述)で登録時刻が記録されていたとすると、何時間前に遡ってでもその時点の在庫は算出可能である。これも在庫ファイルを更新しない利点である。

4．上書き方式と履歴記録方式

すでに存在するデータ

入庫日	原料名	数量
20050401	AEX920	100

上書きの結果 ⇩**80に修正**

入庫日	原料名	数量
20050401	AEX920	80

図３　上書き方式の事例

すでに存在するデータ

入庫日	原料名	数量	登録日時
20050401	AEX920	100	200504021510

履歴記録の結果 ⇩**80に修正**

入庫日	原料名	数量	登録日時
20050401	AEX920	100	200504021510
20050401	AEX920	-20	200504030800

図４　履歴記録方式の事例

データベースの書き換えには上書き方式と履歴記録方式の二方式がある。図３は上書き方式の事例である。データベース内にすでに存在するデータを書き換える。図４は履歴記録方式の事例である。すでに存在するデータはそのまま温存して、差分を新しいレコードを起こして記録する（差分の記録の仕方は図４以外にいくつかのバリエーションがある）。システム安定のためには原則的にマスターデータ、トランザクションデータのいずれも履歴記録方式でなければならない。

5．事実の発生時刻、登録時刻、確定時刻

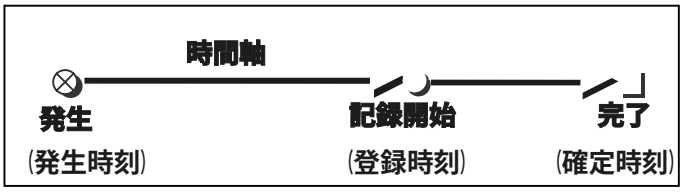

図５　事実の発生時刻、登録時刻、確定時刻

現実の社会で事実の発生から記録までの過程を追うと図５のように、事実が発生して、記録を開始し、記録が完了する。記録の開始から完了までの過程をミクロに見ればもっと細かい様々な事象があるが、これらを知りえても後に利用できないのでこの３つの時刻の記録で充分である。データベースのレコードには図６のように事実の発生時刻、登録時刻、確定時刻の３つの時刻を記録する。発生時刻とは、事実の起こった時刻で、入庫データであれば検印に記されている日時、発注データであれば発注書に書かれている日付、人事データであれば移動日など。但し、事実の発生時刻は省略されることがある。登録時刻とは、データベースに事実を記録するタスクを起動した時刻である。確定時刻とは、データの記録が完了して他のタスクが新しいデータを読むことが可能になった時刻である。

入庫日	原料名	数量	登録時刻	確定時刻
20050401	AEX920	80	200504021510	200504021511

図６　データベースのレコードの事例

6.システム安定のアルゴリズム

事実の発生後、事実を記録するタスクを起動し記録が確定するまでの間、図７のように記録する対象のファイル以外に対象外のマスターファイルやトランザクションファイルなどのデータを参照する。システム安定のアルゴリズムは記録の対象外のファイルのデータの参照時の制約と最終的に事実を確定する制約を主に定めている。その要点は次の通りである。なお、一つのタスクが事実を記録する対象のファイルは一つしかない（ With Single Hand）が前提である。

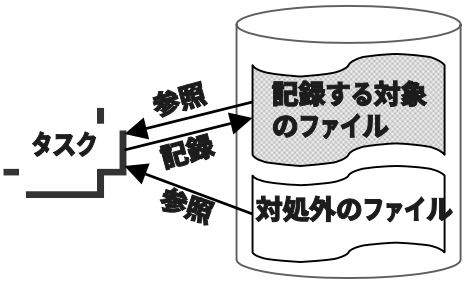

図７　事実の記録のメカニズム

１）タスクが起動された時刻が登録時刻（図６参照）として記録される。
２）タスクが起動された時刻はその後何時間経過したとしてもタスクが終了するまで不変である。
３）タスクが起動された時刻（すなわち登録時刻）はユニークキィを構成する。図６の事例では、「入庫日＋原料名＋登録時刻」がユニークキィとなる。従来は「入庫日＋原料名」である。
４）記録する対象以外のファイルのデータを参照する場合は、参照するレコードの確定時刻を調べてタスクが起動された時刻（すなわち登録時刻）以前のもののみ利用する。同時刻は利用しない。つまり、次の不等式が成立するデータのみが利用対象となる。

タスクの起動時刻（不変）　＞　参照されるレコードの確定時刻

５）データの最新性は登録時刻で判定する。

6）普通はタスクの終了時に記録したデータに確定時刻としてその時点の現時刻を付ける。
7）従来のユニーク・キィ（図６の「入庫日＋原料名」）で検索した結果、新しく登録するデータの登録時刻より大きい確定時刻のレコードが存在した場合、新しいデータは登録できない。

7．実現事項

システム安定のアルゴリズムにより次のことを実現している。
1）記録する対象のファイルは一つしかない（ With Single Hand）こと並びに６の４）の制約により決して書きかえらることの無いデータのみを参照しているので排他制御の必要がない。従って、デッドロックは起きない。
2）あるレコードが記録された時のマスターデータなどの他のファイルのデータはどのようになっていたかを確実に調べることが可能である。誤りの原因や不正記録を追求できる。
3）履歴記録が基本であるので、ある時点に立ち戻ったストップモーションが可能である。
4）マスターデータは全て保存されており、マスターデータとトランザクションデータの前後関係はマスターデータの確定時刻とトランザクションデータの登録時刻をもって明確であるので何年前に遡ってもその当時と同じデータが再現可能である。従って、マスターデータから写し取ったデータをトランザクションデータにも記録しておく必要はない。また特殊なアーカイブ処理も不要である。

8．最新データの定義

図８はタスク１がタスク２により更新されるデータを参照する状況を示している。この図でタスク２で更新されるファイルの最新データはt_1時点のデータであるかt_2時点のデータであるか。従来は排他制御をする/しない、更新時にデータが変わっているかどうかの確認をする/しないで変わった。本論は、更新対象とならないファイルの最新データをt_1時点のデータと定めている。この定め方についてタスク１の終了時にはすでにデータはt_2時点のものになっている以上データ間に不整合が起きるのではないかという懸念が示されている。

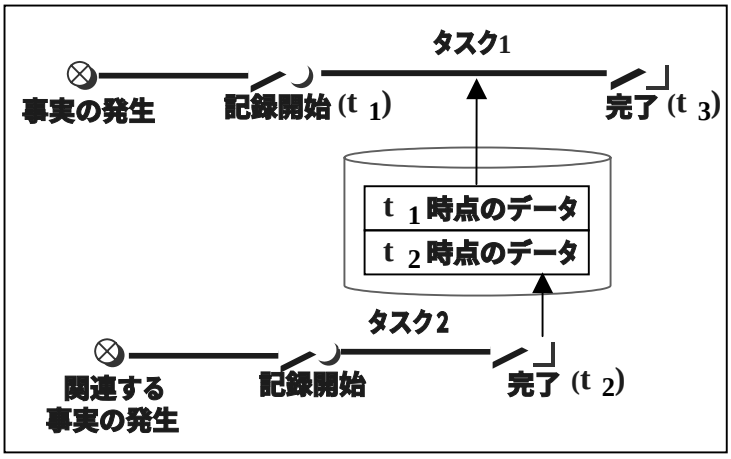

図８　複数タスクによるデータの更新

しかし、t_2 が t_3 より前の時刻である確率は実際上は極小で、逆のことが多いからいずれにせよ問題は起きる。例えば、検収データを登録した後に出荷データが取り消されるようなことがある。こういう場合、どのようにしても矛盾は起きてしまう。従って、With Single Handが守られていてその時点でどのデータを最新であったか分かれば本論の定義で良いと考える。

9．あとがき

企業の営業や生産や物流に従事する人々は、データの記録を待って次の行動を決める従来型の情報システムでは仕事が遅くなることを実感している。ただどうすれば良いかを言えないため致し方なく受け入れているのが現実である。図１に戻って不定時記録の業務に定時記録の方式を適用しようとしてきた従来のアプローチには根本的な無理があったと考える。事実の発生とその記録の間に時間的な乖離があり事実の発生の順番通りの記録が保証されない業務にあって、情報システムは検証可能であることが生命線である。

企業情報システムはとてつもない能力を要求されているのでなく、むしろ、正確な記録の提供が求められている。利用者の要求定義には暗黙的にこのことが含まれている。この暗黙の要求を前提としたソフトウエア・エンジニアリングが必要である。