O-002

オブジェクト指向設計における派生属性のリアルタイム処理化の一検討

A Real-Time Calculation Approach on Derived Attribute Values for Object-Oriented Application Design

川上 拓也† 金田 重郎‡
Takuya Kawakami Shigeo Kaneda

1．はじめに

特定非営利法人技術データ管理支援協会(MASP)が提唱する概念データモデリング（Conceptual Data Modeling，以下 CDM）[1]は，対象ビジネスの構造を写し取り，あるべきビジネスの姿を明確化する手法である．しかし，従来，CDM を適用しても，システム設計に際しては CDM モデルを捨て， RUP などの既存設計手法を適用してオブジェクト設計を行なうことが多かった．その理由は，CDM により得られたオブジェクトが，往々にして「View」のような性格を併せ持ち（例えば，派生属性を含み），データベース構造として最適とは言えないためだと思われる．

しかし，CDM により得られたオブジェクトは，アプリケーション側から見て最適なオブジェクト集合（Persistent Objects）のはずである．従って，これをひとつのレイヤーとしてそのまま実装した方が，アプリケーション側から見て利用しやすく，アプリケーション側における変更要求にも柔軟に対処できる可能性がある．

そこで，本稿では，CDM によるオブジェクトに含まれる集計値などの派生属性を，常に最新の情報に保つ手法について検討する．具体的には，派生属性がもつ属性値を，ソース側オブジェクトが変更されるたびに，リアルタイムに更新することで，CDM により得られたオブジェクト相互の整合性を担保する．実際に，集計処理について，Java によるサンプルコーディングを行なった結果，3 割程度の処理時間増加で，リアルタイム更新が実現できることを確認した．

2．CDM からのデータベース設計

2.1 CDM

CDM は技術データ管理支援協会（MASP）[1]が提案するモデリング手法であり，対象ビジネスにおける情報の流れを取り出して，あるべき情報の流れ（＝ビジネスの姿）を明確化する手法である．CDM には静的モデル，動的モデル，組織間連携モデルなどがあるが，そこで，得られたオブジェクトは，分析に参加したステークフォルダー間で合意された，ひとつの Persistent Object であると考えられる．

2.2 CDMによるオブジェクトのレイヤー化

CDM から得られたオブジェクトを，そのままドメインオブジェクト層とすることで，その上位層（アプリケーション）の設計・保守が容易になることが期待される．ただし，CDM から設計されたデータベースは第 3 正規形に正規化されていたとしても，最小個数のテーブル構成ではなく，派生属性などが含まれる．例えば，ある集計結果が必要だというようなビジネス上の「View」までも取り込んでいることがある．集計値などは処理が重いため，通常のシステム設計では，特定の周期でバッチ処理をするなどして対応している．しかし，このようなアプローチは，アプリケーション側の利用形態を考慮した，下層のオブジェクト設計であり，保守性が劣る．たとえ View 的なオブジェクトであっても，属性値がリアルタイムに更新されていれば，アプリケーション側の設計・保守は容易化すると期待される．

3．商品管理アプリケーションへの適用

本章では以下の CDM 静的モデル[2]（図 1）を参考にし，データベース設計について考える．

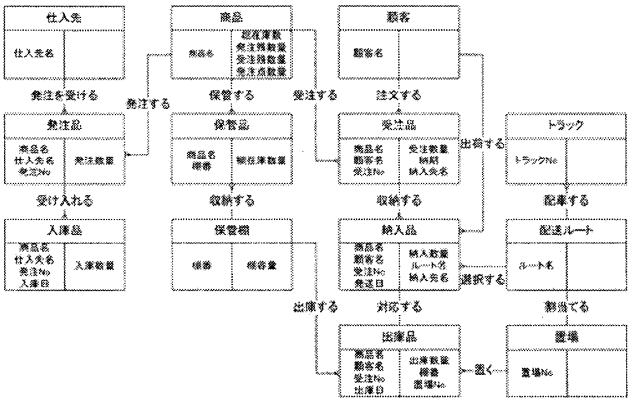

図 1 静的モデル

図 1 において，「在庫品」「入庫品」といったオブジェクトが現れるのは，「在庫品＝在庫＋商品」「入庫品＝入庫＋商品」という人の認識を写像し，「商品」がビジネスにおいて認識されている基本となるカテゴリーであると思われる．商品オブジェクトの識別子が「商品名」となっているが，これは通常ビジネスでは商品はコードではなく名称で識別されているためであり，この商品オブジェクトを ER 図に展開する場合には「商品コード」が主キーとなる．

この「商品」は総在庫数，発注残数量，受注残数量という属性を持っているが，これらは他のエンティティから算出できる派生属性（導出属性）である．この派生属性である総在庫数量の導出には，以下の 2 つの方法がある．

【バッチ処理】例えば，一日の入庫や出庫などの処理が終わった段階で，それぞれの商品ごとの入庫品，出庫品の総合計を総在庫数量に格納してバッチ的な処理で総在庫数量を算出する．処理は通常，深夜となる．

†同志社大学大学院工学研究科
Graduate School of Engineering, Doshisha University

【リアルタイム処理】受注や発注が起こる度に，リアルタイムで総在庫数量をデータベースから取り，差分計算によってアップデートする．これによって，オブジェクトを利用する側は，常に最新の値が属性値であるとして，自由に設計が可能である．

4. 評価実験

実際に派生属性値の集計計算を行い，バッチ処理とリアルタイム処理を行う方法における実行時間を比較する．プログラミング言語：Java，データベース：MySQL，Java と MySQL を接続するための API には JDBC を使用した．あらかじめ商品テーブル，保管品テーブル，保管棚テーブルに任意の値を格納しておき，受注，発注が発生すると入庫，出庫を行うシミュレートを行なった．Main 文に受発注のタイミングや回数を記述し，受発注が発生すると入庫・出庫メソッドを呼び出す，という命令は 2 つの処理方法で共通である．本実験では，CDM（図 1）から「顧客」や「配送ルート」などのオブジェクトを排除した以下の ER 図（図 2）に展開した．

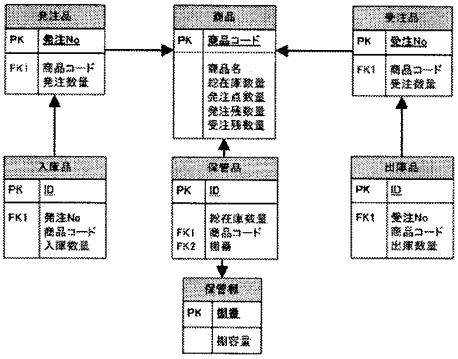

図 2 ER 図

バッチ処理

発注が行われると入庫メソッドが呼び出され，入庫メソッドは発注数を引数とし，その値を入庫数データに格納する．これにより，各商品を発注数だけ入庫したということをシミュレートした．出庫メソッドについても同様である．全ての入庫，出庫が終わった段階で，商品ごとにそれらの値を取り出し，商品項目ごとの総在庫数量の総計を表示させる．バッチ計算は 1 回のみ実施される．

リアルタイム処理

上記と同様，入庫メソッドは発注数を引数として入庫テーブルに格納するが，また同時に，入庫品と同じ商品コードを持つ商品の現在の在庫数を取り出し，その値に入庫数を加えた値を新しい在庫数として，商品テーブルをアップデートする．出庫クラスも同様にして今度は在庫数から出庫数を差し引いた値を新たな在庫数として格納する．バッチ処理と同様，全ての入庫，出庫が終わった段階で商品項目ごとの現在の総在庫数を表示させた．

評価結果

前述のとおり行った結果を次に示す．バッチ処理とリアルタイム処理のそれぞれ 10 個，100 個，1000 個のレコードを作成し，派生属性値を表示するまでに要する時間を示した．バッチ処理で 10，100，1000 個のレコード数で実行した場合，それぞれ 1.6 秒，15.8 秒，162.7 秒の時間がかかり，リアルタイム処理での場合は 2.1 秒，21.4 秒，216.8 秒と計測された．

表 1　実行結果

	10	100	1000
バッチ	1.6 秒	15.8 秒	162.7 秒
リアルタイム	2.1 秒	21.4 秒	216.8 秒

5. 考察

前章の評価実験の結果から，両方法の実行時間は共にレコード数が増えるに応じ，線形的に増えている．受発注が n 回行われたとすると，バッチ処理であれば，受発注命令の受け取り，入庫・出庫の書き込み，最後に総在庫数量の表示のそれぞれは $O(n)$のオーダーで行われているので，実行時間とレコード数が線形関係になる．リアルタイム処理の場合も，同様に考えられるが，この場合，入庫された商品の現在の総在庫数をデータベースから取ってきて逐次アップデートしているので，その分時間を要している．派生属性のリアルタイム処理を行った場合，1 レコードあたり約 1.3 倍の処理時間を要している．

本 CDM の静的モデルには「商品」という抽象的なオブジェクトが出現している．ここでは，当該商品全体の在庫量など，統計的な属性しか興味の対象にはない．一方，それぞれの出荷単位の「商品」は別のオブジェクトにより表現されている．本提案のアプローチでは，アプリケーション側は，どちらのオブジェクトであっても，常に最新の属性値を持っていることを前提にシステムを設計できる．そのための負荷は，発注品などの属性値を更新した場合で CPU 負荷は 1.3 倍に過ぎず，これは，CPU 性能の向上により簡単にカバーできる範囲である．

ただし，本稿では，集計という簡明な処理のみが評価対象である．今後は JOIN により生成された View に相当するオブジェクトが存在する場合など，負荷の重い処理についても，差分計算の性能について評価を行う必要がある．

6. 終わりに

派生属性のリアルタイム処理の場合，差分をとるために，逐次現在の集計値という大きなデータベースを参照する必要があった．このような横断的関心事を AOP(Aspect Oriented Programming)である AspectJ で表現できないかを今後の課題として考えていく．

なお，本稿の議論は，特定非営利法人技術データ管理支援協会の手島歩三氏に示唆をいただいた部分があります．末筆ながら御礼を申し上げます．但し，本稿の内容はすべて著者に責任があることをお断りします．

参考文献

[1]特定非営利法人技術データ管理支援協会(MASP)，Web サイトは次の通り．http://www.masp-assoc.org/
[2]手島歩三ほか「ソフトウェアのダウンサイジング」日本能率協会マネジメントセンター