NoSQL 型データベースシステムでの実体化ビュー選択に関する研究

高木　颯汰†　　古瀬　一隆†　　陳　　漢雄†

† 筑波大学　〒 305-8577 茨城県つくば市天王台 1-1-1
E-mail: †s1511446@u.tsukuba.ac.jp, ††{furuse,chx}@cs.tsukuba.ac.jp

あらまし　本論文では NoSQL の一種であるドキュメント指向データベースに実体化ビューを導入する事によって問い合わせ処理を高速化する手法を提案する．ドキュメント指向データベースでは従来のリレーショナルデータベースにあったような参照型のデータ構造に加えて埋込型のデータ構造を選択できる．参照先の内容を埋め込む事によって結合処理をしなくて済むが，ファイルサイズが大きくなる傾向にあり，フラグメンテーションが発生し，逆にパフォーマンスが落ちる可能性がある．そこで本手法ではリレーショナルデータベースで実現されている実体化ビューの概念を NoSQL にも応用する事で，問い合わせ処理を自動的に高速化する．具体的には，頻繁に問い合わせのある結合処理や集計処理を自動的に検知してその部分のみ予め実体化することでデータベースアクセスの高速化を実現している．

キーワード　NoSQL，Materialized View，Document Database，MongoDB

1　はじめに

数年前までは主要なデータストアとして，リレーショナルデータベースがあげられることがほとんどであった．それは多くの開発者が SQL に慣れ親しんでおり，正規化されたデータモデル，トランザクションの必要性，耐久性のあるストレージエンジンが提供する保証を受けられるからである [1]．しかし近年高いスケーラビリティや大量なデータ処理が得意であることなどから NoSQL に対する需要が急激に増えている．

例えば NoSQL の一種のドキュメント指向データベースはデータベースの構造を表すスキーマを定義する必要がなく，大量なデータを事前準備なしで格納することができる．従来のリレーショナルデータベースにあったような参照型のデータ構造に加えて埋込型のデータ構造を選択できる．型宣言の必要のないスクリプト言語と相性が良いことなども合間って，プロトタイプを高速に開発することが求められるビジネスの現場で採用されることが増えている [2]．

一方でドキュメント指向データベースの特徴とも言える階層的なデータモデルが更新処理速度の低下やデータ参照の柔軟性を低下を招くことがある．これを防ぐためにはドキュメントに階層的に埋め込むフィールドを適切に選択する必要がある．本論文ではこの選択の自動化し，ドキュメント指向データベースのデータモデルのチューニングを行い，データアクセスを高速化する．

本論文の構成は以下の通りである．まず，第 2 章において関連研究について紹介する．次に，第 3 章において本研究の提案手法について説明をし，第 4 章にて提案手法に関する実験を行う．第 5 章において実験の結果と考察を述べ，最後に第 6 章において本論文のまとめと今後の課題を示す．

2　関連技術

2.1　Materialized View

リレーショナルデータベースにおけるビュー (view) はリレーショナルデータモデルの発案者であるコッドにより導入された概念であり [3]，1 つ以上の表（または他のビュー）から任意のデータを選択し，それらを表したものである．ビューの実体はデータを持たない SQL 文であり，実行された際にはバックグラウンドで SELECT 処理が毎回実行される．それに対して実体化ビュー（Materialized View）はビューと同じく複数の表の結合処理や集計処理を行うが，その結果を実際のテーブルに保持する．保持された実体化ビューは元のテーブルが更新されるたびに更新される．そのため，最新でない状態を取得する可能性はあるが，結合処理が必要ないため効率的なアクセスが可能になる．その一方，更新処理が増加するので実体化ビュー化する部分の選択は慎重に行う必要があり，この作業を自動化する研究が行われている [4]．図 1 は実体化ビューを図示したものである．

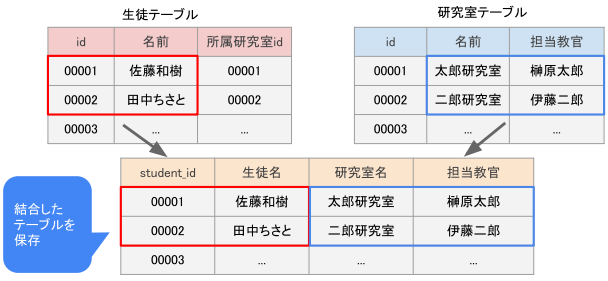

図 1　実体化ビュー

2.2　NoSQL

NoSQL とは，“Not only SQL”の略称であり，SQL を用い

ないデータベースの総称を表す [5]．情報の大規模化が進み，ビッグデータと呼ばれる概念が登場すると共に，構造が複雑な様々なデータが登場するようになった．NoSQL は，そのような複雑な構造のデータに柔軟に対応し処理を行うことができる．Google や Amazon，Twitter など，世界的規模を誇る企業が NoSQL データベースを利用しており，今後ますますデータの大規模化が進む現代社会において，重要な役割を果たすデータベースである [5]．NoSQL データベースはキー・バリュー型，カラム指向型，ドキュメント指向型，グラフ型の 4 種類の型に大別することができる．キー・バリュー型は，インデックスであるキーと値であるバリューのペアでデータが構成され，キーを指定することでデータを呼び出すことができる．カラム指向型は行に対してキーが付され，それが複数の列 (カラム) に対応する形のデータモデルである．ドキュメント指向型は，JSON や XML などの形式で記述されたドキュメントの形でデータを扱うデータモデルである．グラフ型は，データ間の関係性をグラフの構造で表すデータモデルである [5]．

2.3 MongoDB

MongoDB とは，JSON や XML などの形式で記述されたドキュメント指向型のデータを扱う NoSQL データベースの代表的なものの一つである．RDB とは違い，スキーマの定義を必要としない [5] [6]．また，JSON 形式のデータを扱うため，Web システムなどに利用しやすい．MongoDB においては，RDB のテーブルにあたるものとしてコレクション，RDB の行にあたるものとしてドキュメント，RDB の列にあたるものとしてフィールドというデータ構想が使われる．ドキュメント指向型データベースの特徴として埋め込み（embed）がある．従来の RDB では複数の表による 1 対多や多対多の関係を表す際に，参照先のプライマリーキーのみを保存して SELECT される際に結合処理を行う．それに対してドキュメント指向型データベースでは参照先の実データを参照元に埋め込むことができ，これによって結合処理を省くことができる．埋め込み先が複数の場合には更新処理が増加し，従来の参照型に比べてデータアクセスの柔軟性が損なわれるというデメリットがある [1]. 図 2 はドキュメント指向型データベースの参照型を埋込型を表した図である．

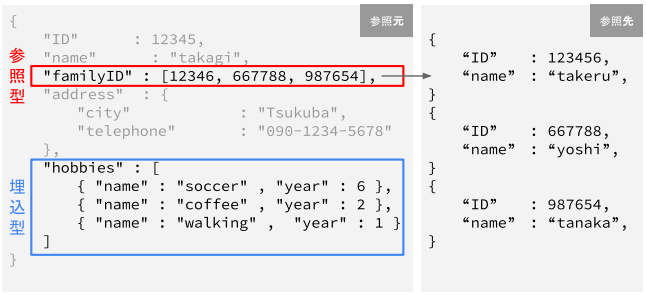

図 2　参照型と埋込型

2.4 Restful API

REST とは Roy Fielding が提唱した概念であり [7]，“REpresentational State Transfer”の略である．分散システムにおける複数のソフトウェアを連携させるのに適した考え方であり，やりとりされる情報はそれ自体で完結して解釈できるステートレス性，全てのリソースが一意的なアドレスを持つアドレス可能性，他の基盤的な機能を用いずに別の情報や状態を含むことで他のリソースを参照できる持続性，HTTP メソッド（“GET”や “POST” など）の統一インターフェースを提供していることなどの原則から成る．REST の原則に則り構築された HTTP の呼び出しインターフェースを RESTful API と呼ぶ．本論文では RESTful API をミドルウェアに実装し実験を行う．

3 提 案 手 法

3.1 Mongoose について

Mongoose とは MongoDB 用モデリングツールで，Node.js の非同期環境でうまく動作することを目的として設計されている．Mongoose を使用すれば，モデルを定義して操作することで，MongoDB のコレクション/ドキュメントを操作できる [8]．本論文では Mongoose を用いて MongoDB を操作するミドルウェアを実装する．

3.2 実装システムについて

本論文ではドキュメント指向型データベースの埋込型データモデルをリレーショナルデータベースの実体化ビューと置き換えて考える．ドキュメント指向型データベースのよくアクセスされる部分や処理速度がネックとなっている部分を埋込型として別コレクションに保持することで “実体化ビュー作成”，どの部分を埋込型にするかの判断を “実体化ビュー選択” とする．図 3 は本論文での “実体化ビュー作成” を示したものである．

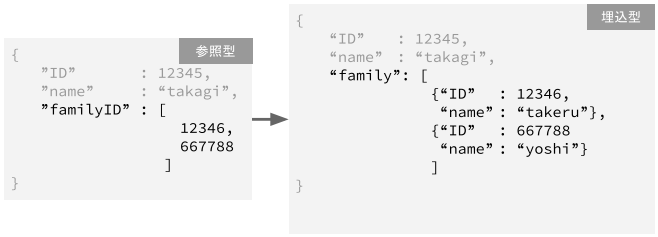

図 3　参照型から埋込型への書き換え

実装システムについて図 4 に示す．ユーザーからのデータアクセスから実体化ビュー作成までの流れを図 4 を用いて説明する．図 4-①まずユーザーがアプリケーションからミドルウェアに対してデータアクセスの要求する．図 4-②ミドルウェアでは，頻繁にアクセスされるドキュメントを分析するために，クエリに関するログを残す．図 4-③次に MongoDB に対してクエリを発行する．図 4-④ MongoDB から返ってきたクエリセットをアプリケーションに返却する．図 4-⑤クエリログを解析し，ボトルネックとなっているところや呼び出し回数の多い条件の実体化ビューを作成する．図 4-⑥実体化したドキュメン

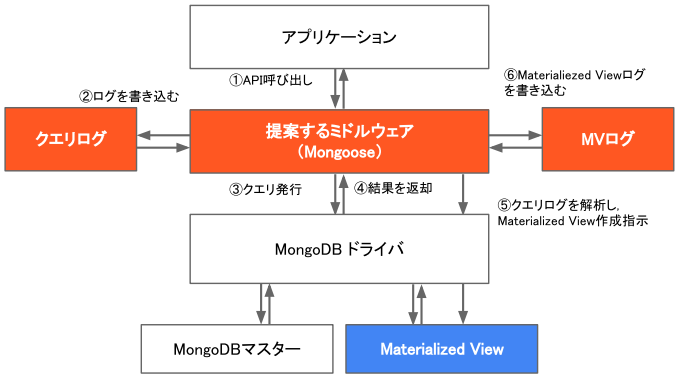

図 4　提案ミドルウェア（実体化前）

トに関してログに記録する．

実体化した後のデータアクセスの流れを図 5 に示す．図 5-①’アプリケーションからデータベースにアクセスがあった場合，まずログからアクセスされたデータが実体化されているか判定する．図 5-②’実体化されている場合はクエリを書き換えて実体化ビューから結果を取得する．図 5-③’アプリケーションに結果を返す際には元のクエリに合うように適宜変換する．

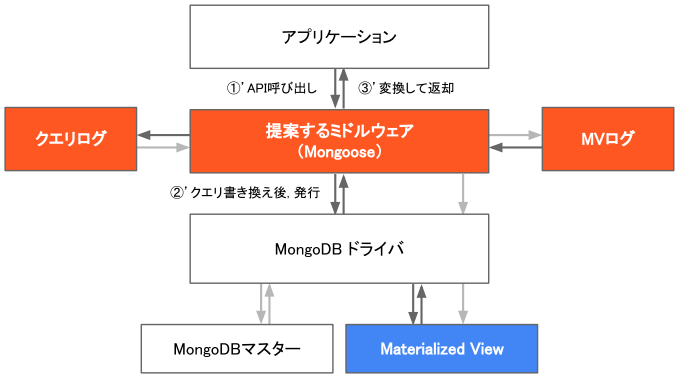

図 5　提案ミドルウェア（実体化後）

4　実　　験

5　結果・考察

6　ま　と　め

7　謝　　辞

本研究を進めるにあたり，指導教員の古瀬一隆先生と陳漢雄先生から，丁寧かつ熱心なご指導を賜りました．ここに感謝の意を表します．また，研究室での議論を通じ，多くの知識をいただいた DSE 研究室の皆様に感謝いたします．

文　　献

[1] Kyle Banker. “MongoDB イン・アクション”. 第 1 版, 株式会社オイラリー・ジャパン, 2012, 356p.
[2] 渡部徹太郎. “RDB 技術者のための NoSQL ガイド”. 第 1 版, 株式会社 秀和システム, 2016, 549p
[3] E. F. Codd, “Recent Investigations in a Relational Database System,” Information Processing 74, pp. 1017 - 1021, NorthHolland, 1974.
[4] Hoshi Mistry, Prasan Roy S, Sudarshan, Krithi Ramamritha. “Materialized view selection and maintenance using multi-query optimization”, Proceedings of the 2001 ACM SIGMOD international conference on Management of data, pp. 307-318, 1-58113-332-4, 2001.
[5] 本橋信也, 河野達也, 鶴見利章. “NOSQL の基礎知識 ビッグデータを活かすデータベース技術”. 第 1 版, 株式会社リックテレコム, 2012, 255p.
[6] “What is MongoDB?”. mongoDB. https://www.mongodb.com/what-is-mongodb, (参照 2019-01-09).
[7] Fielding, R.T. Architectural Styles and the Design of Network-Based Software Architectures. Ph.D. Thesis, University California, Irvine, CA, USA, 2000.
[8] “mongoosejs”. mongoosejs. https://mongoosejs.com/ （参照 2019-01-09）