1ZC-02

製品地図システムにおける MySQL と MongoDB の性能比較評価

古川友貴† 工藤司†
静岡理工科大学 情報学部 情報デザイン学科†

1. はじめに

多品種の製品を製造する工場では、あらかじめロット単位で製造した在庫を保管し、注文に応じて出荷する方法が一般的である。しかし、1000 種類に及ぶ製品を保管する場合には、担当範囲が限定されるため、担当者不在などの際には在庫管理に支障が発生することがある。

そこで、誰でも製品の場所を把握できることを目的として、製品の画像や製品番号などの情報を組み合わせ、製品が保管されている棚ごとに地図を作成する棚地図システムを開発した。製品はサイズや形状、製造工程の場所に基づき、工場内に分散して保管されており、地図もエリア、棚、製品のように階層化して作成した。

しかし、実際の運用では、画像管理が煩雑なために、画像のデータベース(DB)化と、受注ごとの製品情報が必要になった。そこで、本研究では製品地図システムを提案すると共に、データ操作と画像操作を対象に、MySQL と MongoDB の性能比較評価結果を示す。

2. 製品地図システムの提案

注文は仕様番号(ID)で特定される仕様書で指示され、仕様書には仕様 ID のほかに製品番号(ID)、注文した製品数が指示されている。そこで、DB に製品画像と出荷に必要な情報を登録し、対応する仕様 ID を入力することで保管場所(棚 ID)、製品 ID、製品の在庫数、製品画像を検索する製品地図システムを提案する。

図 1 に本システムのデータの流れを示す。注文があった時、注文に対応する仕様 ID を入力し DB を検索する。そして、検索した上記の出荷情報と画像を出力フォルダに出力する。画像の名称は、「棚 ID_製品 ID_製品の在庫数」に変更することで、画像と製品の情報が把握できるようする。さらに出力した画像をタブレットに転送し、現場で参照できるようにする。

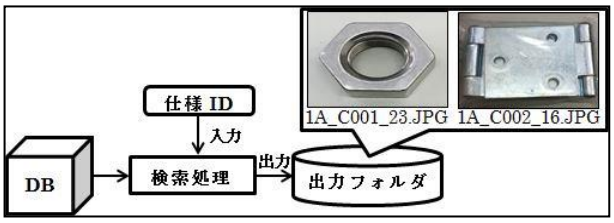


図 1：製品地図システムのデータの流れ

3. データモデルとデータベース選定上の課題

図 2 に提案システムの ER 図を示す。仕様書、製品テーブルが各々のマスタであり、製品在庫数の履歴が在庫状況テーブルに格納される。さらに、在庫状況テーブルのから棚 ID、製品 ID ごとに、最新更新日のデータを抽出した最新在庫ビューを作成する。また、在庫履歴に対応した製品の画像が在庫画像テーブルに格納される。本システムでは、仕様 ID を指定し、仕様書、製品、最新在庫ビューを結合して出荷情報を検索し、該当する画像を検索する。

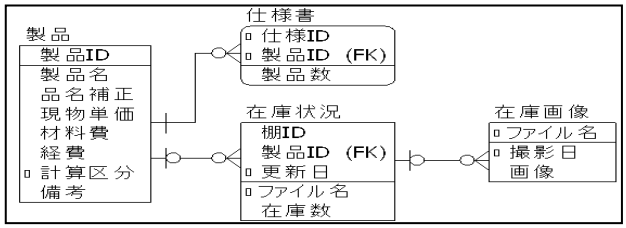


図 2：製品地図システムの ER 図

このように、本システムでは結合演算と画像データ操作が必要である。そこで、DB の選定のため、従来のリレーショナルデータベース(RDB)である MySQL と大容量データ操作に優位性がある MongoDB の性能評価を行う。

MongoDB はドキュメント指向の NoSQL DB であり、データモデルは図 3 に示す木構造で表現できる。先行研究では、動画などの大容量データ処理に適していることを確認した[1]。しかし、RDB と比較して結合演算やビューが提供されていないという課題がある。

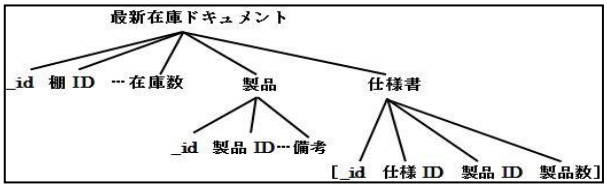


図 3：MongoDB(非正規形)の構造

この課題に対し、非正規化したドキュメント（MySQL のレコードに相当）を持つ MongoDB のデータモデルを併せて評価する。このデータモデルでは、あるドキュメントに他のドキュメントを組み込む。本研究では、ドキュメントの容量の増大を防ぐため、データ数の多いコレクション（同じく、テーブルに相当）のドキュメントに組み込むこととする。図 2 の ER 図に対しては、図 3 に示すように、最も多い最新在庫のドキュメントに製品と仕様書を組み込む。ここでは、最新在庫と製品は多対 1 であるため、製品の各属性をそのまま組み込む。一方、最新在庫と仕様書は 1 対多であるため、各々の仕様書の属性集合を図 3 に[ ]で示す配列として組み込んだ。

本データモデルでは、1 つのコレクションに全てのデータが格納されているため、結合演算が必要ない。なお、上記の仕様書の配列では、検索条件として指定した仕様 ID を含むドキュメントが検索対象になる。

以上の点から、本研究では表 1 に示す 3 つのデータモデルを対象に性能比較評価を行う。表 1 で、「△」は課題があることを示し、「－」は対象外であることを示す。

Comparative Evaluations between MySQL and MongoDB in Product Map System
† Yuki Furukawa † Tsukasa Kudo
† Shizuoka Institute of Science and Technology

また、MongoDB ではビューは提供されていないため、検索の都度、同様の操作を実行した。

表 1：対象 DB の性能における予想

	MySQL	MongoDB(正規形)	MongoDB(非正規形)
結合(Join)	○	△	-
画像	△	○	○

4. MongoDB の実装

本研究では、Java 8、MySQL Ver.5.6.17、MongoDB Ver.3.4.6 を使用して実装し、MySQL ドライバは、Ver.5.1.44、MongoDB ドライバは Ver.3.5.0 を使用した。

最新在庫ビュー作成には在庫状況の自己結合が、出荷情報検索には 3 節の結合演算が必要である。そこで、MySQL ではプログラムで SQL 文を作成し、ドライバによって実行した。

MongoDB は結合とビューの機能がないため、Java プログラムで作成する。ビューは次の手順で最新在庫コレクションを作成した。まず、在庫状況から棚 ID と製品 ID でグループ化し、更新日の最大値を得る。その結果のリストで在庫状況を順次検索し直し、対応するファイル名と在庫数を取り出した。結合に関しては、仕様書から検索対象のキーをリストで抽出し、製品を順次検索する。その結果抽出されたリストで最新在庫を順次検索することで、3 つのコレクションから任意のデータを取り出した。MongoDB(非正規形)では、1 つのドキュメントに全てのテーブルのデータを挿入しているため、結合演算を使用せずに検索のみを実行した。

在庫画像は、まず、他のテーブルのデータ操作により、キーとなる「ファイル名」を取得し、画像データを検索した。取得したファイル名から画像データを出力する操作の実装を図 4 に示す。例えば、MySQL では select into dumpfile 命令を使用することで画像を出力する。一方、MongoDB では、GridFS インターフェイスが提供されており、Java のストリーミングを定義し、gridFSBucket クラスで ObjectID により画像を出力する。ここで、ObjectID はファイル名から検索できる。

```
(1) MySQLの挿入
insert into 在庫画像 values ( '棚ID_製品ID_在庫数' ,' 20170706' , load_file( "ファイル名" ));
(2) MySQLの検索
select 画像 into dumpfile '棚ID_製品ID_在庫数' from 在庫画像;
(3) MongoDBの挿入
InputStream st = new FileInputStream(new File("ファイル名"));
ObjectId fileId = gridFSBucket.uploadFromStream("棚ID_製品ID_在庫数", st);
(4) MongoDBの検索
FileOutputStream st = new FileOutputStream("ファイル名");
gridFSBucket.downloadToStream(fileId, st);
```

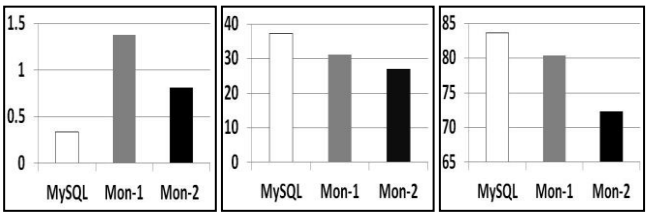
図 4：画像データ出力の実装

5. データベースの性能比較評価

DB の評価のために、単体性能として、仕様書、製品、最新在庫を結合し、必要なデータを検索するまでの処理時間を評価した。評価は PC のスタンドアロン環境で実施し、CPU は Intel Core i5-2320 (3.00GHz)、メモリは 4.0GB、OS は Windows7 である。条件は指定せず、結合した全ての行(240 件)を対象とした。ここで MongoDB(非正規形)は結合を使用しないため、コレクションからデータを直接検索した時間になる。結果を図 5 の(1)に示す。ここで、Mon-1 は MongoDB(正規形)、Mon-2 は MongoDB(非正規形)である。MySQL が 0.3 秒、Mon-1 は 1.4 秒、Mon-2 は 0.8 秒であり、MySQL が MongoDB(正規形)に比べ約 4 倍速く処理できた。

次に、1MB から 4MB の画像計 120 枚を登録して出力するまでの時間を評価した。その結果、MySQL は 37.3 秒、MongoDB(正規形)は 31.2 秒、MongoDB(非正規形)は 27.1 秒かかり、画像の入出力に関しては MongoDB が有効であった(図 5 の(2))。

その上で総合性能として、注文に対応する仕様 ID を入力し、出力する画像を検索し、特定のフォルダに出力するまでの処理時間を計測した。ここでは、実際に工場で使われている約 1000 個の製品画像とデータを使用している。その結果、MySQL は 83.6 秒、MongoDB(正規形)は 80.4 秒、MongoDB(非正規形)は 72.3 秒であり、MongoDB(非正規形)が最も早い時間で処理することができた(図 5 の(3))。



(1)結合　(2)画像入出力　(3)総合性能

図 5：単体性能、総合性能評価結果

6. 考察

図 5 に示すように、非正規形の MongoDB が最も速く処理できることが分かった。これは、結合演算が不要であるため、大容量データに対する MongoDB の優位性を生かせたことによる。また、今回はマスタデータのため、データ更新が少なく、更新時異常が発生しない。このように、マスタデータに関する画像の操作に関しては、MongoDB の非正規形が有効だと分かった。

また、図 5 の(1)と(2)に示すように、結合演算の時間に対して、画像操作の時間は非常に長い。すなわち、非正規化が難しい場合であっても、MongoDB を適用することにより性能を向上することができる。さらに、4 節に記載したように Java などのプログラム言語を使用することにより、画像のデータ操作と、画像ファイル名の取得を含む他のデータ操作を分離できる。従って、画像データの保管、操作の部分に MongoDB を適用する構成もまた有効であると考えられる。

7. まとめ

MySQL と MongoDB には、各々、優位な部分があり、業務システムへの適用には選定上の課題が発生する。本研究では、画像操作と結合演算を含むシステムである製品地図システムを対象として、両者の性能比較評価を行った。この結果、画像操作を含む場合には MongoDB が有効であることが分かった。さらに、非正規形や MySQL との補完など、対象業務応じた構成とすることが重要であることが分かった。なお、本研究は JSPS 科研費 15K00161 の助成を受けたものです。

参考文献

[1] 長澤一貴, 工藤司：タブレットと MongoDB を活用したモバイル見積システムの開発, 信学会総合大会講演論文集, 情報・システム１, p. 113 (2017).