健康診断データ後利用システムの開発と評価 ～ユニケージ開発手法によるパフォーマンス改善～

藤井 香[*1] 大貫 亮[*2] 松本 可愛[*1] 高橋 綾[*1] 清 奈帆美[*1] 櫻井 勉[*1] 金子 康樹[*2]
押見 淳[*2] 河邊 博史[*1]

[*1]慶應義塾 大学保健管理センター [*2]慶應義塾 インフォメーションテクノロジーセンター

Development and evaluation of health checkup data later-use system -Performance improvement by Unicage software development method-

Fujii Kaoru[*1] Onuki Akira[*2] Matsumoto Kaai[*1] Takahashi Aya[*1] Sei Nahomi[*1]
Sakurai Tsutomu[*1] Kaneko Yasuki[*2] Oshimi Atsushi[*2] Kawabe Hiroshi[*1]

[*1]Health Center, Keio University [*2]Keio Information Technology Center

The health checkup system in our university uses general RDBMS(Relational Data Base Management System) and has a complicated table structure containing plural medical examination types and inspection data each year. Since it is a structure with which it is difficult to data-link by relation of simple keys and items such as those for which single IDs are based on single reception information, handling of mainframe search performance is extremely slow and it affects our services. Moreover, in data addition, update and deletion by SQL, system call for securing memory is issued each time and garbage collection is performed as its nature and therefore it worsened the performance even more. In the Unicage software development method (R), data are in text format, which realizes relation by linking files with shell script, and it is not necessary to secure memory during the process repeatedly. From an experiment that shell script was executed on a multi-core processor / STM machine, it has been reported that the processing speeded up by 1.4-1.8 times compared with that before applying the method. In this way, we have built a data later-use system using Unicage software development method and therefore aim at evaluating it. In file generation (binding logic) for all checkup data, which are used by a data later-use system, processing time for approximately 30,000 students' health checkup data of one year has been improved to approximately ten seconds. In addition, for mainframe search, as a result of comparing all items' output for which CSV output capability was used from RDBMS and extracting logic in the data later-use system for their processing time, it has been revealed that the output processing time for the subject was approximately 5,700 seconds for the former and approximately 1.1 seconds for the latter, suggesting that the processing time was vastly improved. This implementation of this time has made us expect promotion of efficiency in performance aspect in services and further data utilization in the study.

Keywords: Unicage software development method, High speeding technique , SQL

1. はじめに

当大学における健康診断（健診）システムIDST(Information and Database of health care Service Tools)は、2011年より教職員健診、学生健診、特定業務従事者健診、雇入時健診の4健診で利用している。

IDSTは一般的なRDBMS（Relational Data Base Management System）を利用し、ユーザの要望にて、単年度単位で複数の健診種類や検査データを含む複雑なテーブル構造を持つ。単純な受付機能やデータ入出力機能には問題はないが、単一IDが単一の受付情報に紐づくような、単純なキーや項目のリレーションでデータ連携することが難しい構造であるため、汎用検索の処理パフォーマンスは極度に遅く、業務に支障をきたしていた。かつ、SQL（Structured Query Language）によるデータ追加、更新、削除は、その度にメモリ確保するシステムコールを発行し、ガベージコレクションを行う性質から、よりパフォーマンスを悪化させていた。

ユニケージ開発手法（Unicage software development method）®[注1)][1,2]は、データをテキストファイル形式とし、シェルスクリプトでファイルを連携させることによってリレーションを実現でき、処理中何度もメモリ確保を行う必要が無い[3,4]。マルチコアプロセッサ・SMT（Surface Mount Device）マシン上でシェルスクリプトを実行した実験[5]では、手法適用前に比べて1.4～1.8倍の速度向上が可能であったと報告がある。そこで今回、ユニケージ開発手法を用いて、データ後利用システムを構築したので評価したい。

注1）ユニケージ開発手法

UNIX系オペレーティングシステム上（特にLinux上）において、シェルスクリプトのみでシステムを開発する手法。通常のテキストファイル形式でデータを持たせ、独自開発されたフィルタコマンド群をパイプラインで接続して並列処理する技術を用いる。

2. 対象と方法

業務用ネットワーク上での健診システムにおけるデータ後利用システムとして、結合／抽出ロジック[注2)]のパフォーマンスを中心としたシステム全体の評価を行った。

IDSTを利用した検索・全項目CSV出力と、今回開発した抽出ロジックの機能を用いた同一データの検索・

CSV出力における処理時間を計測し、パフォーマンスの比較を行った。結合／抽出ロジックにおいては、2006年度～2011年度各年度の教職員健診、学生健診の検索・全項目CSV出力時間、2～6年度分の複数年度での検索・全項目CSV出力時間（秒）を計測した。

なお、IDST、結合／抽出ロジックそれぞれの処理に用いたサーバはその用途やシステムの特性上の観点から仕様上異なるが、IDSTで利用した基幹サーバのマシンスペックは、2012年度学生健診実施時におけるパフォーマンス監視結果でみると、CPU使用率は最大約20used%であり、十分なリソースの余力があったことを確認している。このことから、処理時間の差は、データ構造上の問題、及びそれに付随したアプリケーションの実装上の差であり、基幹サーバのマシンスペック上の問題とは考えにくい。

注2）

結合ロジック;

出力されている複数の検査データから年度及び健診種類ごとに一　受付一レコード（≒一人一レコード）にマージする機能。

抽出ロジック;

結合ロジックにてマージしたCSVファイルを対象に、WEBクライアントからの任意の条件によるリクエストに対して、ヒットしたデータをCSVファイル形式で出力する機能。

3. システム概要

3.1 システム形態（図1.WEB3階層と処理フロー）

システム全体を、ユーザーインターフェイス層（クライアント側）、ビジネスロジック層（サーバ側）、データベース層（サーバ側）の3階層に処理を分割して構成するWEB3階層型とした。

本システムでは、①ユーザーインターフェイス層をCSV出力用画面（クライアントPC上のWEBブラウザ）、②ビジネスロジック層を結合／抽出ロジック（サーバ上のシェルスクリプト）、③データ層を各検査項目データ（サーバ上のCSVファイル）の3つの要素で3階層とした。

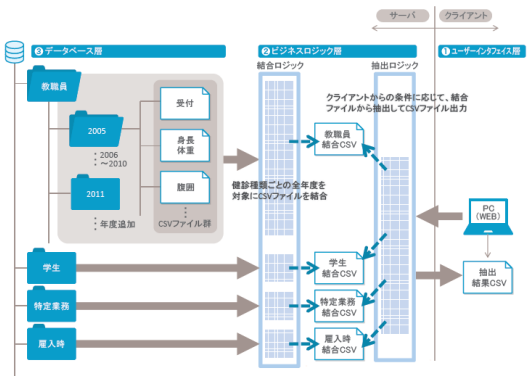

図1　WEB3階層と処理フロー

3.2 結合／抽出ロジック　サーバ構成

結合／抽出ロジック専用環境として独立したマシン上に構築した。

3.3 ハードウェア

IBM System x3550 M3（1台）
CPU：インテル(R)Xeon(R)プロセッサE5620(4コア/8スレッド)2.40GHz x2　メモリ：48GB(4GB x12)
ディスク：300GB(内蔵型SAS)　x4、実効容量600GB(RAID10構成)

3.4 OS／ミドルウェア

サーバOSとしてCent OS（フリーのLinuxディストリビューション）バージョン6を利用した。さらに、本システムのコアとなるusp Tukubaiコマンド群®（有限会社ユニバーサル・シェル・プログラミング研究所）をインストールした。なお、リレーショナルデータベース管理用のミドルウェアは利用していない。クライアントPCからのWEBアクセス用には、OS標準のApache HTTP Server（オープンソース・ソフトウェア）を利用した。

3.5 パフォーマンス要件

全データの結合処理および各種抽出処理について、複数年度にまたがる処理であっても5分以内に終了することを目標値とした。

3.6 データ構造と結合／抽出ロジック（図2）

プログラムは、usp Tukubaiコマンド群を利用したシェルスクリプトのみで記述した。

IDSTのデータ構造は、健診種類ごとにテーブルが存在し、同一年度、同一健診種類でも、一人に紐付くべきデータが不定数レコード存在しており、検査ごとにCSV出力する機能を持っている。そこで、結合ロジックは、IDSTから出力された各検査項目のCSVを対象に、年度及び健診種類ごとに一受付一レコード（≒一人一レコード）にマージする処理を実装した。

また、抽出ロジックは、結合ロジックにてマージしたCSVファイルを対象に、WEBクライアントからの任意の条件によるリクエストに対して、ヒットしたデータをCSVファイル形式で出力する機能を実装した。データ結合ロジックの仕様として、同一健診内における複数回検査による列ずれが発生しないように、また、結合対象となる各検査項目データについては、個人属性など同じ内容の列があった場合に、それらも重複しないように制御した。結合のキーとなるのは健診受付日および各検査日であり、同一人物が同一種類の健診を二度受診したとしても、複数回検査や外部受診についても直近の健診に紐付くように設定した。受付日のないデータは不要データの位置付けで精査され、元となる健診システムへのフィードバックを可能とした。

3.7 クライアント利用

業務用PC標準のWEBブラウザであるInternet Explorer9（日本マイクロソフト株式会社）での利用を想定した。

3.8 ネットワーク・セキュリティ

ファイアウォールによる制限が設けられている業務用LAN（有線）を利用した。また、サーバ側でのアカウント管理、およびシステム利用時の認証機構と、使用端末登録による利用制限を設けた。

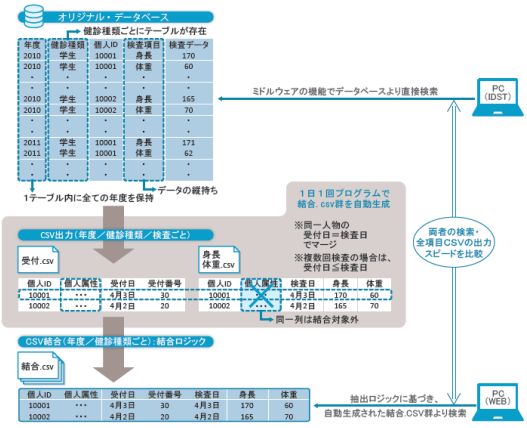

図2 データ構造と結合／抽出ロジック

4. 結果

4.1 IDST、抽出ロジックによる検索・CSV出力のパフォーマンス評価（表1）

データ後利用システムで用いる全ての検査データについてのファイル生成（結合ロジック）では、約30,000人の学生健診データ1年度分の処理時間が約10秒であった。また、RDBMS(IDST)のCSV出力機能を利用した検索・CSV出力と、抽出ロジックの処理時間の比較を行ったところ、2009年度学生健診の場合、出力処理は、前者がおよそ5,700秒、後者がおよそ1.1秒という結果であった。2009年教職員健診でみると、前者が1,500秒、後者が0.341秒であった。

表1 IDST、抽出ロジックによる検索・CSV出力のパフォーマンス評価

健康診断システム(IDST)におけるCSV処理時間の計測

教職員健診	件数	1回目	2回目
2008 年度	5,575	1,440.000	1,380.000
2009 年度	5,780	1,500.000	1,500.000

（秒）

学生健診	件数	1回目	2回目
2008 年度	28,687	5,100.000	5,040.000
2009 年度	29,709	5,700.000	5,700.000

（秒）

抽出ロジックにおけるCSV処理時間の計測

教職員健診	件数	1回目	2回目
2006 年度	5,130	0.272	0.272
2007 年度	5,302	0.295	0.309
2008 年度	5,575	0.329	0.332
2009 年度	5,780	0.341	0.360
2010 年度	5,834	0.335	0.354
2011 年度	6,228	0.382	0.388

（秒）

学生健診	件数	1回目	2回目
2006 年度	27,546	1.006	1.083
2007 年度	27,761	1.015	1.029
2008 年度	28,687	1.124	1.172
2009 年度	29,709	1.187	1.096
2010 年度	30,124	1.180	1.244
2011 年度	30,131	1.217	1.362

（秒）

結合ロジックにおけるCSV処理時間の計測（参考）

教職員健診	件数	1回目	2回目
2006 年度	5,130	2.404	2.383
2007 年度	5,302	2.460	2.520
2008 年度	5,575	3.190	3.256
2009 年度	5,780	3.244	3.334
2010 年度	5,834	3.032	3.111
2011 年度	6,228	3.441	3.385

学生健診	件数	1回目	2回目
2006 年度	27,546	8.167	8.317
2007 年度	27,761	8.242	8.232
2008 年度	28,687	8.521	8.512
2009 年度	29,709	8.809	8.758
2010 年度	30,124	10.208	10.262
2011 年度	30,131	10.668	10.302

（秒）

4.2 複数年度を対象とした、抽出ロジックによる検索・CSV出力のパフォーマンス評価（表2）

抽出ロジックを用いた検索・CSV出力の処理時間を2～6年度分の複数年度で計測した。教職員健診では、2006～2011年度(6年度分)33,847件1.9秒、学生健診では、2006～2011年度(6年度)173,958件6.6～6.7秒であった。

5. 考察

今回実装したユニケージ開発手法を用いた後利用システムは、IDSTに比較し、圧倒的に処理時間が短く、パフォーマンスが優位であることが示された。ユニケージ開発手法のシステム構成上の特徴と優れた点として、データ管理におけるミドルウェア＝データベースを必要とせず、プログラムにおける複雑な記述が発生しないことが挙げられる。プログラムはusp Tukubaiコマンド群を用いたシェルスクリプトのみ、データはテキストファイルのみで構成されるという非常にシンプルで無駄の無い構成である。また、ユニケージ開発手法はコマンドをOSのパイプで連結してデータストリームの並列処理を行わせる方式で、処理目的にのみコンピュータリソースを有効に使うので、実行速度が著しく速く、開発当初の処理時間目標値をはるかに上回る結果が達成できた。職員の業務負担になることなく、さらなる業務面での効率化や蓄積されたデータの後利用として研究面での発展も期待できた。

また、システムの特性上、CSVファイルを非常に容易に扱えることで、他システムとの連携が柔軟である。今後の診療系システム、画像系システム、医事会計システムなどの導入に応じてそれらシステムとの連携を通じ、保健管理センターとしての健診結果への利活用だけでなく、診療データや日常の健康記録も併せたビッグデータ分析にも有用となることであろう。

ユニケージ開発手法はミドルウェアを利用しないため、不要なランニングコストやアップデート作業などが発生せず、データもプログラムもテキストファイル形式なので、今後のハードウェアの入れ替えなどにおいても単純なファイルコピーだけでシステム移行が済むことから、将来的な管理面での負荷も軽減されることが期待された。

一方で、データ自体がデータベースではなくファイルで管理されているという特徴から、サーバ自体へのアクセス制限と権限設定などセキュリティに関しては、他のシステム同様に十分な配慮が必要である。

6. 結語

ユニケージ開発手法を用いたデータ後利用システムのプログラム実行処理は著しく速く、SQLでの処理時間目標値をはるかに上回る結果が達成でき、従来システムが抱えていたパフォーマンスの問題は改善された。データの二次的な活用として、ビッグデータ分析にも有用だと思われた。

※ ユニケージはユニバーサル・シェル・プログラミング研究所の登録商標。
※ usp Tukubaiはユニバーサル・シェル・プログラミング研究所の登録商標。

表2 複数年度を対象とした、抽出ロジックによる検索・CSV出力のパフォーマンス評価

数職員健診	件数	1回目	2回目
2006-2007 2年度分	10,430	0.508	0.533
2006-2008 3年度分	16,005	0.815	0.843
2006-2009 4年度分	21,785	1.185	1.188
2006-2010 5年度分	27,619	1.547	1.519
2006-2011 6年度分	33,847	1.859	1.856

(秒)

数職員健診	件数	1回目	2回目
2007-2008 2年度分	10,876	0.619	0.585
2007-2009 3年度分	16,656	0.899	0.905
2007-2010 4年度分	22,490	1.416	1.211
2007-2011 5年度分	28,718	1.577	1.623

(秒)

数職員健診	件数	1回目	2回目
2008-2009 2年度分	11,355	0.653	0.686
2008-2010 3年度分	17,189	1.031	0.972
2008-2011 4年度分	23,417	1.382	1.389

(秒)

数職員健診	件数	1回目	2回目
2009-2010 2年度分	11,614	0.733	0.666
2009-2011 3年度分	17,842	1.289	1.024

(秒)

数職員健診	件数	1回目	2回目
2010-2011 2年度分	12,062	0.680	0.712

(秒)

学生健診	件数	1回目	2回目
2006-2007 2年度分	55,307	1.998	2.009
2006-2008 3年度分	83,994	3.079	3.060
2006-2009 4年度分	113,703	4.392	4.052
2006-2010 5年度分	143,827	5.464	5.370
2006-2011 6年度分	173,958	6.559	6.583

(秒)

学生健診	件数	1回目	2回目
2007-2008 2年度分	56,448	2.099	2.086
2007-2009 3年度分	86,157	3.273	3.158
2007-2010 4年度分	116,281	4.553	4.419
2007-2011 5年度分	146,412	5.650	5.398

(秒)

学生健診	件数	1回目	2回目
2008-2009 2年度分	58,396	2.230	2.169
2008-2010 3年度分	88,520	3.377	3.378
2008-2011 4年度分	118,651	4.561	4.571

(秒)

学生健診	件数	1回目	2回目
2009-2010 2年度分	59,833	2.329	2.316
2009-2011 3年度分	89,964	3.562	3.809

(秒)

学生健診	件数	1回目	2回目
2010-2011 2年度分	60,255	2.455	2.402

(秒)

参考文献

[1] ユニケージ開発手法. http://www.usp-lab.com/methodology.html.

[2] UECジャーナル ユニケージ開発手法とは. https://uec.usp-lab.com/INFO/CGI/INFO.CGI?POMPA=ABOUTUNICAGE.

[3] 片山善夫,當仲寛哲. Universal Shell Programming (USP)向けシェル機能と実装(シェルプログラミング・Webアプリケーション). 情報処理学会研究報告. ソフトウェア工学研究会報告 2008(112), 47-54, 2008-11-12.

[4] 松岡泰史,當仲寛哲. ユニケージ手法 : 高生産性で安価な顧客重視の開発手法(シェルプログラミング・Webアプリケーション). 情報処理学会研究報告. ソフトウェア工学研究会報告 2008(112), 55-62, 2008-11-12.

[5] 杉田秀, 他. マルチコア・SMTプロセッサ上におけるシェルスクリプト高速化手法(コンパイラ及びツール,「ハイパフォーマンスコンピューティングとアーキテクチャの評価」に関する北海道ワークショップ(HOKKE-2007)). 情報処理学会研究報告. 計算機アーキテクチャ研究会報告 2007(17), 73-78.