Vol.2015-DBS-162 No.24
2015/11/26

自己結合 SQL クエリ検出ツールによるチューニングの提案

岸本拓也[†1]　坂本一憲[†2]

概要：非機能要求の一つとしてパフォーマンス要件があるが，パフォーマンス確認手段は負荷テストによって評価されている場合が多い．パフォーマンスが充分でない場合の向上策として，データベース上の表アクセス数を減らすことで I/O 量を削減し，SQL クエリの応答時間を短縮することが挙げられる．本研究では複数表にアクセスしている SQL クエリから，書き方を工夫すれば表アクセスを 1 回に削減できるにも関わらず，同一表に対してアクセスしている SQL クエリを定義し，検出するツールを開発した．本ツールを稼働中の商用システムにて評価したところ 95%の精度，96%の再現率で検出することができた．

キーワード：データベース，パフォーマンスチューニング，SQL

A Method for Detecting Self-join SQL Queries

TAKUYA KISHIMOTO[†1]　KAZUNORI SAKAMOTO[†2]

Abstract: Performance requirement, which is one of the non-functional requirements for enterprise environment, tend to be evaluated by stress test. We can improve the response time our applications and queries by reducing the number of unnecessary I/O operations. An important facet of database system performance tuning is the tuning of SQL queries. We developed the tool of detecting self- join SQL queries which can rewrite a query. When we evaluate this tool on the large-scale production environment, the results indicate that it can detect self-join SQL queries, which reach 95%.

Keywords: Database, Performance Tuning, SQL

1. はじめに

ビッグデータは企業システム内における普及期の時代に入り，企業が扱うデータベースサイズは巨大化が進んでいる．データベースのパフォーマンス性能は I/O 量と I/O スピードが大きく作用している．データベースサイズが大きくなれば，システムパフォーマンスを維持するために様々なチューニング策を検討する必要がある．

非機能要求の 1 要素として挙げられるパフォーマンス要求[1]であるが，実際の開発現場において開発早期段階に要件定義が検討され，検証されていないことがある．開発後期に実施される負荷テストでの結果から，データベースに対してチューニングを実施するとなると表の構造的な見直しなどは現実的にはできない．例えばデータベース正規化の見直しは，変更の影響が広範囲にわたり，手戻りコストが多くかかってしまい採用することが困難である．

本研究はデータベースサイズを本稼働時と同等サイズに確保できていない小規模な単体・結合テスト段階であっても，パフォーマンスが問題になるような SQL クエリの書き方を見つけることを目的とする．そこで，SQL クエリの修正のみで改善でき，改善の効果が高く，変更の影響範囲を最少に抑えられるようなパフォーマンスの問題箇所を調査した．実際の開発現場で調査を行った結果，負荷テストでパフォーマンスの問題が見つかり，人手をかけてチューニングを行っている SQL クエリは表結合数が多いものが含まれていることが分かった．チューニングが必要な SQL クエリの多くに，表結合の処理が含まれている原因は（1）複雑な結合条件を理解し，更に仕様を理解することが困難であり，どのようなチューニングを施すべきか判断が難しい．（2）アクセスパスを決定するオプティマイザが，総当たりの結合条件集合の中から最適なアクセスパスを導出できていないことの 2 点が挙げられる．表結合を含む SQL クエリは，全ての結合対象の表が異なるとは限らず，同一表を繰り返し結合している自己結合 SQL クエリの場合がある．自己結合 SQL クエリの書き方を工夫して，1 回の表アクセスで目的の結果セットを全て取得できれば，I/O 量を削減でき，パフォーマンスを改善できる．以上の調査を踏まえ，本研究では単体テストや結合テスト時に実行された SQL クエリを対象として，書換えることで I/O 量を削減できるような自己結合 SQL クエリを SQL 文のパターンマッチングにより，検出するツールを開発した．

本稿は 2．節で自己結合 SQL クエリの例と SQL クエリ書換えによる応答時間の改善方法例を紹介する．3．節では作成プログラムの仕様を述べる．　4．節ではある商用環境

†1 株式会社インサイトテクノロジー
Insight Technology, Inc.
†2 国立情報学研究所
National Institute of Informatics

から抽出した SQL クエリに本ツールを適用した実証実験結果を紹介する．5. 節では関連研究について触れる．最後に本研究のまとめについて述べる．

2. 自己結合 SQL クエリと改善方法

SQL は書き方の自由度が高い言語である．同じ結果セットを得る SQL クエリでも様々な書き方が許容される．SQL1 はある表から id 別の最大値を含む特定行を抽出する SQL クエリであるが，表に対して 2 回アクセスしている．I/O アクセス数も 2 倍となるので表サイズが大きければ，応答時間に深刻な影響を与えることもある．SQL1 は WINDOW 関数[2]を利用することで表アクセスを 1 回に削減した SQL2 に書換えが可能である．1 億行の表検索で比較したところ，SQL1=3.94 秒，SQL2=2.66 秒の応答時間となった．このように SQL クエリの書き方を変えることで表アクセス数を削減し，応答時間を削減することができる．

```
select id,val1,val2 from tab1 as a, (select id,max(val1) as val1 from tab1 group by id) as b where a.id=b.id and a.val1=b.val2;
```

SQL 1 自己結合 SQL クエリ
SQL 1 Sample of Self-join SQL Query

```
select id,val1,val2 from (select id,val1,val2, row_number() over(partition by id order by val1 desc) row_number from tab1) where row_number=1;
```

SQL 2 自己結合 SQL クエリのチューニング例
SQL 2 Method of Tuning Self-join SQL query

3. 作成ツール概要

本研究で作成したツールは，作成した自己結合 SQL クエリのモデルが，実行されている SQL クエリに含まれているかどうか成否判定する．データを本稼働時の想定サイズと同等量準備し，応答時間を測定し，パフォーマンス要件を満たすかどうかの負荷テストは，開発の最終工程に行われることが多い．本ツールは SQL クエリの書換えによってチューニングが可能かどうか判別できる．データサイズによるパフォーマンス測定のみに頼らず，テストデータが揃っていない単体テストや結合テスト時に書換えによってパフォーマンス向上が可能な SQL クエリを抽出できる．

3.1 自己結合 SQL クエリのモデル化

業務における経験則として SQL クエリ書換えによりパフォーマンス向上が見込める以下 3 パターンの自己結合 SQL クエリを検出モデルとした．モデルは SQL クエリで記述する．テキストファイルとして保存し，ツールがモデルとして取込む．

1）特定列の最大値・最小値を結合条件とした自己結合
例: Select col1, .. from tab1 join (select max(col1) from tab1) on ..
2）特定列の値を結合条件とした自己結合
例: Select col1, .. from tab1 where tab1 col1 = (select col1 from tab1)
3）UNION による自己結合
例: Select col1, .. from tab1 union select col1, .. from tab1

3.2 自己結合 SQL クエリの判定方法

書換え可能な自己結合 SQL クエリモデルが，データベース上で実行された SQL クエリに適用可能かどうかの判別には，構文解析を行ったうえで表・列名のデータベース固有データを取り除いた SQL クエリ構造が同一かどうかで判定する．構文解析器には ANTLR[3]を用い，解析結果を Python で扱いやすい XML ファイルに変換する（図 1）．Python コードは XML パーサモジュール[4]を利用し，モデル定義をリスト構造に変換した XML タグリスト（図 2）および要素データの全てが検査対象 SQL クエリに同順序で含まれる場合，書換え可能な自己結合 SQL クエリとして成否判定をする（図 3,4,5）．以下に自己結合 SQL クエリのモデル化手順を示す．

Step1）データベースから実行された各 SQL クエリを各テキストファイルとして抽出
Step2）SQL クエリテキストファイルから ANTLR により，構文解析
Step3）構文解析結果から XML ファイルに変換
Step4）モデル化定義した SQL クエリファイルを上記 2), 3)手順により，XML ファイルに変換
Step5）各 XML ファイルの TOKEN タグを含むパスツリーを抽出
Step6）XML の要素情報から表，列名などの個別情報を特定の名前に変換
Step7）モデル化定義した XML タグパスツリー，要素データ全てが検査対象 SQL クエリに同順序で含まれる場合，書換え可能な自己結合 SQL クエリとして判定する

```
<parse id="-1">
  <sql_stmt_list id="164">
    <sql_stmt id="182">
      <factored_select_stmt id="225">
        <select_core id="641">
          <K_SELECT id="128">
            <TOKEN id="128" startline="1"
startpos="0" endline="1"
```

図 1 XML 出力例
Figure 1 XML Output Sample

```
['', 'parse', 'sql_stmt_list', 'sql_stmt', 'K_SELECT', 'TOKEN', 'SELECT']
```

図 2 XML タグリスト例
Figure 2 XML Tag List Sample

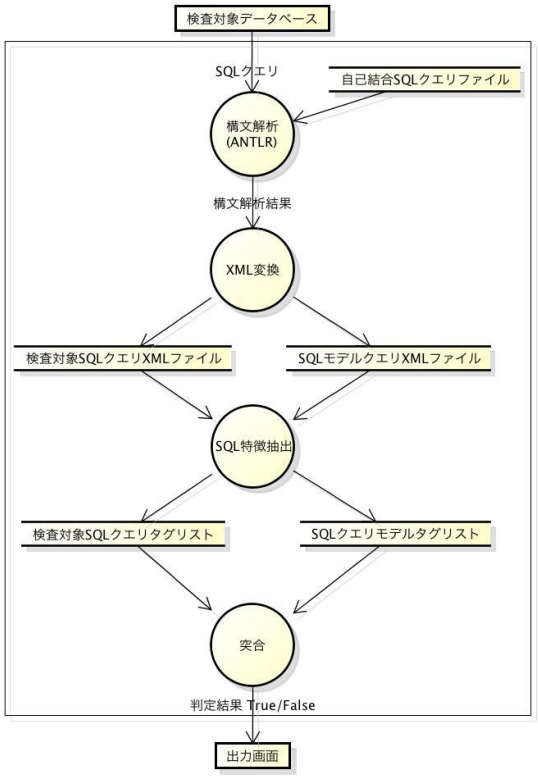

図 3　書換え可能な自己結合 SQL クエリ判定フロー
Figure 3 Method of Detecting Self-join SQL queries

```
Function XML から SQL クエリ特徴抽出
  import XML パーサ as xparse
  #初期化
  xtree = [] #各 XML パスツリーを格納
  xlist = [] #各 XML タグ・要素を格納
  #変換処理
  xtree = xparse.XML ファイル
  xlist = xparse.xtree #2 次元リスト
  #処理
  for xtag_list in xlist
  if TOKEN タグ in xtag_list
      表・列名を特定名に変換
      不要タグを削除
      SQL 特徴 list = xtag_list
Function SQL クエリ特徴リスト比較
  突合 list = 自己結合 SQL_list in 検査対象 SQL_list
  if 突合 list == 自己結合 SQL_list
```

図 4　作成したツールのアルゴリズム
Figure 4 Detecting Tool Alogorithm

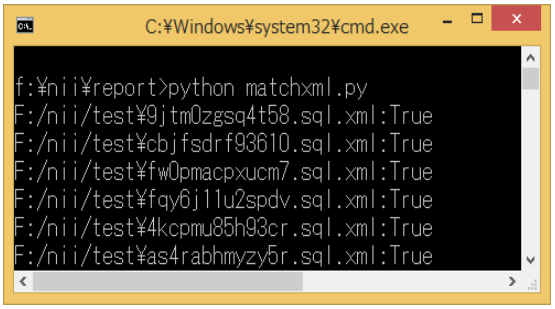

図 5　自己結合 SQL クエリモデル検出ツール出力画面
Figure 5 Detecting Tool Output Screenshot

4. 評価実験結果

作成ツールが自己結合 SQL クエリを正しく抽出できるか妥当性検証を行うため，Oracle のデータベース上に構築されている 1 つの商用環境から実行計画を確認し，自己結合 SQL クエリと非自己結合 SQL クエリをそれぞれ抽出した．それぞれの SQL クエリを XML 変換ツールにより，XML ファイルに変換した後に本ツールを実行し，精度検証を行った．

4.1 自己結合 SQL クエリの抽出精度検査

まず，自己結合されている SQL クエリを実行計画で確認し，132 抽出した．このうち 127（96%）がモデルに合致した．モデルに合致しなかった 5SQL クエリは，View の使用，OR 検索，In 検索により実行計画上では自己結合がされていたが，SQL クエリでは表結合をしていなかった．

4.2 誤検出の精度検査

次に非自己結合 SQL クエリを同一環境から 261 抽出し，誤検出の検証を行った．12SQL クエリが誤検出された．誤検出の原因はツールのロジックに原因があると考えられる．今後原因究明のうえ，修正を施し再テストの実行をしたい．

	自己結合 SQL クエリ	非自己結合 SQL クエリ
検査対象件数	132	261
検出できた件数	127	-
誤検出件数	5	12

表 1 ある商用環境下における評価実験結果
Table 1 Evaluation Result on Production Environment

再現率	精度
96%	95%

表 2　評価実験評価
Table 2 Accuracy of Tool

5. 関連研究

パフォーマンス改善を目的とした書換え可能な SQL クエリの検出に関する研究はいくつかある．

Araújo らが開発した ARe-SQL はヒューリスティックな 11 のモデルにより，書換え可能な SQL クエリの検出を行い，高いパフォーマンス改善結果を残している[5].

Karwin は著書の中で 25 の SQL クエリアンチパターンを定義し，その中の 1 パターンとして自己結合をしているような複雑な SQL クエリを Spaghetti Query として紹介している[6].

商用データベース用の製品も存在する．Belknap らは Oracle11g の機能である SQL Tuning Adviser について紹介している[7]．SQL Optimizer for Oracle[8]は書換え代替 SQL クエリを表示する機能を実装している．Embarcadero DB Optimizer XE[9]は書換え可能な SQL クエリの修正案を表示する．Optim Query Tuner for DB2 for Z/OS は書換え候補の SQL クエリを表示する[10].

先行研究ないし，製品では書換え可能な SQL クエリの定義はプログラムコードに組み込まれているため，SQL クエリモデルの追加はツールのコード修正の必要がある．一方本研究で作成したツールは SQL クエリモデルを追加することができる工夫をした．モデルはテキストファイルに SQL を記述する形式とした． Python コードを扱えなくても，SQL クエリを習熟している DBA やアプリケーションエンジニアなどの技術者であれば，書換え可能な SQL クエリをテキストファイルとして作成をするだけで，本ツールが表・列名情報を自動的に消し込んだうえでモデルとして追加される．また，自己結合 SQL クエリに着目した研究および製品は存在しない.

6. まとめ

本研究において書換えることでパフォーマンス改善が可能な自己結合 SQL クエリを抽出するツールを開発した．非手続型言語である SQL クエリは同一結果セットを得るコードでも様々な書き方が可能である．開発現場でチューニング対象になる SQL クエリの多くは単純な SQL クエリはあまりなく，複数表結合している複雑な SQL クエリであった．複数表結合している SQL クエリは検索対象表全てが違う表ではなく，同一表を結合（自己結合）している場合がある．自己結合 SQL クエリは，書き方を変更し，自己結合を解消すれば表アクセス数を削減し，応答時間を削減することができる．本研究にて開発したツールは，自己結合 SQL クエリのモデルとして定義したテキストファイルを読込み，構文解析したのち，SQL クエリ構造をリスト型に変換し，検査対象の SQL クエリとの比較を行い，合致成否結果を出力する．商用環境における実証実験から 3 パターンの自己結合 SQL クエリモデルに合致する多くの SQL クエリを検出できた．本結果から作成したツールの妥当性を実証できた．適合率の精度向上は今後の課題とする．今後，他の商用環境でも本ツールを適用し，単体・結合テスト時においてもパフォーマンス改善に貢献する有用なツールとして開発現場で利活用できると考えている．様々な用途のデータベースや様々なベンダーのデータベースに本ツールを適用することで，ツールの精度は更に向上し，書換え可能な SQL クエリモデルも自己結合 SQL クエリに留まらず，増えていくと考えている．

パフォーマンスチューニングは様々な要素を検討する必要があり，実際に効果があるチューニング方法を導き出すことは困難である．開発後期や本稼働後にチューニングを施すとなると，なるべくコードの内容を変えずにチューニングする必要があるので，SQL クエリにヒント句を挿入したり，本研究と同様に SQL クエリの書換え程度しかできない．本ツールを適用することで本来チューニングが必要である SQL クエリが開発早期に検出され，問題になる前にパフォーマンスを考慮した SQL クエリを書換えられるとよい.

参考文献

[1] 非機能要求グレード. http://www.ipa.go.jp/sec/softwareengineering/reports/20100416.html 2015/10 参照
[2] Eisenberg, Andrew, et al. "SQL: 2003 has been published." ACM SIGMoD Record 33.1 (2004): 119-126.
[3] Terence Parr. The definitive ANTLR 4 reference. The Pragmatic Bookshelf, 2012.
[4] lxml. http://lxml.de 2015/10 参照
[5] de Araújo, Arlino HM, et al. "ARe-SQL: An Online, Automatic and Non-Intrusive Approach for Rewriting SQL Queries." Journal of Information and Data Management 5.1 (2014): 28.
[6] Bill Karwin. SQL Antipatterns: Avoiding the Pitfalls of Database Programming. Pragmatic Bookshelf. 2010.
[7] Belknap, Pete, et al. "Self-tuning for SQL performance in Oracle database 11g." Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on. IEEE, 2009.
[8] Dell. SQL Optimizer for Oracle http://software.dell.com/products/sql-optimizer-for-oracle/. 2015/10 参照
[9] Embarcadero. Embarcadero DB Optimizer XE. https://www.embarcadero.com/jp/products/db-optimizer. 2015/10 参照
[10] IBM. Optim Query Rewrite for DB2 for Z/OS http://www-03.ibm.com/software/products/ja/optiquertunefordb2forzos 2015/10 参照