最適化プロファイルを利用したアクセス・プランの固定化設計

大月 真史　下総 哲郎

Design to Stabilize the SQL Access Plan by Using an Optimization Profile

Masafumi Otsuki and Tetsuroh Shimoosa

ダイナミック・インフラストラクチャーには，ビジネスの変化を俊敏に先取りして変革し続ける柔軟性と共に，一過性の変化に対して過度に反応せず，先見性を備えた姿に固定化することでサービス品質を維持し，不要なシステム運用コストを削減し，リスクを増加させないための制御技術もまた，極めて重要となる．近年，オープン系のデータベースがミッション・クリティカルな基幹系業務に浸透するに伴い，データベース処理の性能管理がますます重要となってきているが，性能管理の重要な要素であるアクセス・プラン管理において，固定化技術も含めたアクセス・プランの制御方法はこれまで確立していなかった．筆者らは，DB2® for Linux®，UNIX®，Windows®（DB2/LUW）の機能である「最適化プロファイル」を用いたアクセス・プランの固定方法について設計と実装を行い，現時点の最適化プロファイルが持つ課題を独自の工夫によって解決し，大規模システムへの適用に成功した．本論文では，ダイナミック・インフラストラクチャーを実現するために必要となる安定性を制御する技術の事例として，DB2/LUW のアクセス・プランの固定運用をする上での設計と考慮点について論述する．

Dynamic Infrastructure requires not only flexible systems which enable proactively and prompt response to rapid business changes, but also control engineering which allows stabilization of the system, carefully designed to accommodate future needs. By stabilizing the system with control engineering technology, we are able to maintain the service quality level with minimum risk and maintenance costs by not reacting excessively to temporary situations. In recent years, it is becoming more and more important to control database query performance as many mission critical systems adopt open-system RDBMS. Controlling access plans is one of the key factors for performance management. However, there have been few comprehensive methods to control the access plan in “DB2 for Linux, UNIX and Windows” (DB2/LUW). In our recent system integration project with a large scale DB2/LUW database, we designed and implemented a method to stabilize the access plan by an optimization profile, which is the latest function of DB2/LUW. We have been able to solve the essential consideration points of optimization profiles with our newly devised implementation when we put them into a large scale production database system. In this paper, we will describe our design and points to consider when applying the optimization profile to a large scale production system as an example of using a unique way of controlling stability that supports the implementation of a dynamic infrastructure.

Key Words & Phrases：最適化プロファイル，アクセス・プラン，性能管理
Optimization Profile, Access Plan, Performance Control

1. はじめに

近年，オープン系の大規模データベースが，基幹系システムに対して本格的に求められるようになり，データベースの性能を設計段階から向上させるために，アクセス・プラン［1］［4］の最適性を追求する取り組みが行われている．また，DB2/LUW はラージ表スペースやパーティション表など，1 つの表で数百 TB に達する巨大表を構築可能な機能を提供してきたが［2］，取り扱うデータが巨大になるほど，アクセス・プランの予期せぬ変化がパフォーマンスへ及ぼす影響は大きくなる．そのため，大規模データベースにおいて SQL のアクセス・プランを最適化し，変更可能な形で固定化する手法は，ますます重要となってきている．

提出日：2008年9月8日　再提出日：2009年6月1日

ホスト・システムにおいては，静的 SQL を採用し，十分なデータ量を確保した上で統計情報の更新を停止したり，あるいは，DB2 for z/OS® の機能である OPTIMIZATION HINT［3］を使用したりすることによって，アクセス・プランを固定化している．しかしながら，DB2/LUW をはじめとするオープン系データベースでは，下記の理由によって，ホスト・システムと同様の運用形態が，必ずしも適切とは限らない．

- 基幹系のオンライン・トランザクション処理（OLTP）と，情報系の複雑な SQL 処理の両方を処理することが求められるケースが見られる．データ量や分布の変動を伴う情報系の複雑な SQL 処理では，実態と異なる統計情報は性能劣化の原因となりやすいため，統計情報を更新しない方法の採用は困難である．
- DB2/LUW では，統計情報の自動更新機能や，統計情報の一部を推測する機能などが次々と導入されている．そのため，統計情報更新を行っていないにもかかわらず，パッケージの再バインドのタイミングで異なるアクセス・プランが採用されてしまうケースもあり，DB2/LUW では統計情報更新を停止してアクセス・プランを固定化する運用は，実用的ではなくなりつつある．

筆者らの携わる DB2/LUW による基幹系大規模データベースを構築するプロジェクトにおいても，性能管理のためアクセス・プランの変動を抑止することが求められ，下記のような設計方針を取った．

- 静的 SQL を使用するため，オンライン処理では Java™ プログラムから静的 SQL が利用できる SQLJ を，バッチ処理では組み込み SQL を採用
- DB2/LUW の独自機能である最適化プロファイルを採用し，アクセス・プランを固定
- 意図したアクセス・プランが採用されていることを確実に確認するため，EXPLAIN［1］［4］情報を基にモデルとなるアクセス・プランと一致しているかどうかをチェックする機能を構築

これらの設計方針を採用したことにより，DB2/LUW では困難であったアクセス・プランの確実な固定が可能となった．当プロジェクトでは，最適化プロファイルの採用を基本設計の時点から決定し，効率的なアクセス・プランの選定とその固定をタスク化し，実際のプロジェクトの設計フェーズから統合テスト・フェーズに適用した．その結果，すべてのアクセス・プランを意図した形に固定することに成功した．

日本国内において DB2/LUW の最適化プロファイルを実運用されるシステムに適用した事例は，サービスイン後のパフォーマンス・トラブルの緊急避難的な改善のため，ごく一部の SQL に対して手作業で実施されたものが数件あるのみである．アクセス・プランの安定化を目的として設計段階から採用し，本番運用に組み込む形で実用化した事例やノウハウは数少ない．本論文では，最適化プロファイルの採用とアクセス・プランの同一性を確認する仕組みの構築において，筆者らの担当するプロジェクトにおける経験に基づき，設計と考慮点を提示することを目的とする．

2. 最適化プロファイルとその設計

2.1 最適化プロファイルとは

最適化プロファイルは，アクセス・プランを制御する機能で，DB2/LUW V8.2 から使用可能となった．しかし，当初は公式に提供される情報がなく，実際の使用どころか，機能を理解することすら困難であった．DB2 V9.1 よりようやくマニュアルへの記載が行われ［4］，IBM の開発部門からの情報［5］なども提供され始めたが，本論文で記載するような，実システムへの適用に当たっての課題とその解決について述べた資料は，いまだ十分とはいえない．

最適化プロファイルは，XML フォーマットを用いたテキスト・ファイルであり，制御対象の SQL や指定するアクセス・プランを XML ファイル中に記述し，データベース内の表へ登録することで使用可能となる［6］．アクセス・プランの制御は，表へのアクセス方式や結合方式といった，性能管理上極めて重要な部分のみを対象としている．安定した性能を提供することが必須条件として求められる大規模基幹系のデータベースにおいて，統計情報の変動により結合方式や，内部表と外部表の選択，使用さ

```
<?xml version="1.0" encoding="UTF -16"?>
<OPTPROFILE VERSION="9.1.00">
<!-- Global optimization guidelines section. Optional but at most
  one. -->
<OPTGUIDELINES>
</OPTGUIDELINES>
<!-- Statement profile section. Zero or more.  -->
<STMTPROFILE ID="SQLID3">
  <STMTKEY SCHEMA="PROF"><![CDATA[SELECT
 LIST1.FLG ,LIST1.CONTRACT_NO ,LIST1.CREATE_DATE ,LIST1.CREATE_SEQ
 _NO ,LIST1.INPUT_DATE FROM  DETAIL_LIST1 LIST1 ,MASTER1  MST1
 WHERE LIST1.FLG=MST1.FLG AND LIST1.CONTRACT_NO
 =MST1.CONTRACT_NO AND MST1.CUSTMER_NO      ='?']]>
  </STMTKEY>
 <OPTGUIDELINES>
  <NLJOIN>
   <IXSCAN TABLE='LIST1'  INDEX='IX0DETAIL_LIST1'/>
   <IXSCAN TABLE='MST1'   INDEX='IX1MASTER1'/>
  </NLJOIN>
 </OPTGUIDELINES>
 </STMTPROFILE>
</OPTPROFILE>
```

指示対象のSQLを記述

指示するアクセス・パスを記述

図1. 最適化プロファイル XML例

れる索引が変動することは性能管理上大きなリスクとして挙げられる．このような大きなリスクを低減させるための手段として，最適化プロファイルは強力な機能を有する．

図１に最適化プロファイルの XML 例を示す．この例で示すように，最適化プロファイルは対象の SQL そのものと，その SQL に対して指示するアクセス方式の組み合わせを基本単位とする．また，一つの XML ファイル中に複数の SQL に対するアクセス・プランの指示を含むことができる．SQL ステートメント・レベルのアクセス・プランの指示は，STMTPROFILE タグ中で指定し，ここにアクセス方式や結合方式を記述する．

図２に，最適化プロファイル適用前後のアクセス・プランの変化を示す．ここでは図１に示したアクセス・プランを適用しており，結合方式がハッシュ結合（HSJOIN）からネスト・ループ結合（NLJOIN）へと，表へのアクセス方式が表スキャンから索引スキャンへと変わっていることが読み取れる．この最適化プロファイルの例では，基礎表からの結合方式と使用する索引のみ指定し，それ以外のアクセス方式については DB2 による最適化を許容している．

例えば，図２右側のアクセス・プランは，最適化プロファイルが適用された状態であっても，MASTER 表からの索引スキャンの後ソートが行われる場合がある．これは，オプティマイザーが処理対象の行が多いと判断した場合に，ネスト・ループ結合による DETAIL_LIST 表のアクセスを効率化するために行われる．

このように，最適化プロファイルは，DB2 のオプティマイザーによるアクセス・プランの最適化を許容しつつ，パフォーマンスの安定化のためにアクセス・プランのキーとなる要素を固定することが可能とする．

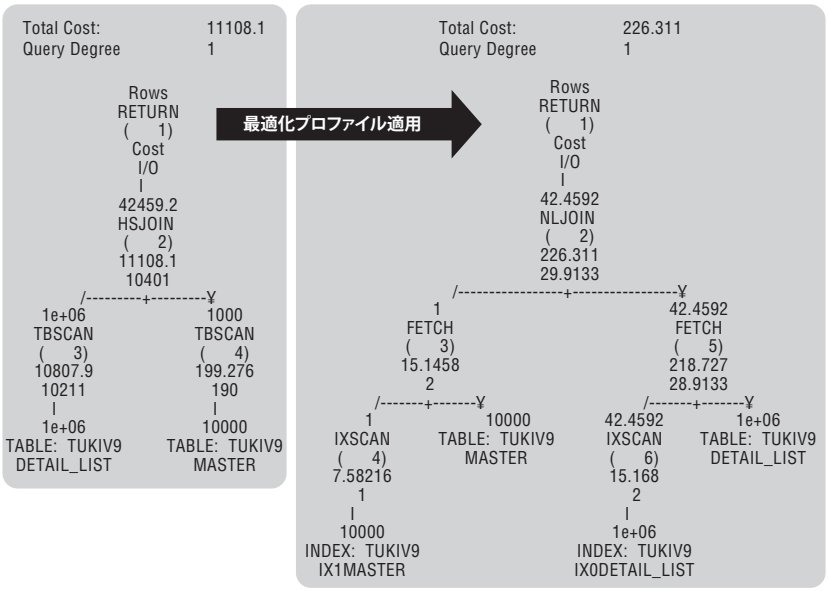

図2. 最適化プロファイルによる制御の例

2.2 実運用上の課題点

最適化プロファイルはアクセス・プランの固定化を行う上で強力な機能を有するが，現時点で提供されている標準機能では，下記のような重大な課題点が存在するため，大規模な実運用には適用できない．

課題１ SQL と最適化プロファイル間での厳密な同期の必要性

実行される SQL と最適化プロファイルをひも付けるため，最適化プロファイル内に SQL の全文を記述する必要があり，大文字小文字の食い違いも許容されない．そのため，例え軽微な修正であっても，SQL が変更されるたびに最適化プロファイル中の SQL 差し替え，XML ファイルの再作成，データベースへの登録が必須となる．

課題２ 最適化プロファイルの汎用的利用が困難

最適化プロファイルの XML ファイル中に，表や索引の名称を正確に指定する必要がある．そのため，同じ類型の SQL であっても，個別に最適化プロファイルを作成する必要がある．

課題３ アクセス・プラン妥当性チェックの困難さ

最適化プロファイル適用後，意図するアクセス・プランになっているかどうかを確認する資料は，db2exfmt［7］ツールによる EXPLAIN 情報のみである．db2exfmt は，アクセス・プランに関する情報を極めて豊富に出力するので，意図したアクセス・プランになっているかどうかを，簡易なスクリプトで自動判定することは困難である．目視によるアクセス・プランの確認が再 BIND のたびに必要となるので，SQL 数が多いシステムでは膨大な労力と時間を要する．

大規模システムでは，類似する SQL や表が多数存在するのが常である．筆者らの担当したプロジェクトでは，約 32 億件（800GB）の巨大な表を分割し，同じ構造の中規模表（5,000 万～ 1 億 5,000 万程度）を約 40 表保持する設計であったため，同じ類型の SQL が多数存在した．それぞれの SQL ごとに最適化プロファイルを手動で作成し管理することは管理ワークロード上現実的ではなく，何らかの自動化が必須となった．次章では，筆者らが独自に創出したアクセス・プランを自動的にチェックする仕組みを含む，最適化プロファイルの自動運用について説明する．

2.3 実運用を考慮した適用フロー

図３に，SQLJ における最適化プロファイル使用時の一般的な作業フローを示した．このフローでは，最初に ser ファイルを基に SQL の EXPLAIN 情報を取得し，取得した SQL をベースとして最適化プロファイルの XML ファイルを編集する（工程 1 および工程 2）．作成された XML ファイルを SYSTOOLS.OPT_PROFILE 表に INSERT することでデータベースへの登録が完了する（工程 3）．その後，登録したプロファイルを指定して再 BIND を行うことで，最適化プロファイルの適用を完了させ（工程 4），結果を EXPLAIN 情報にて確認する（工程 5）との，5 つの工程から成っている．

この作業フローの中で，工程 2 および，工程 5 が多大なワークロードを要する部分であり，前述した課題 1 から 3 が発生するポイントである．

2.4 最適化プロファイル生成自動化の骨子

前述の作業ワークロードを要する工程を最小化するために，筆者らは図４に示す最適化プロファイル生成の自動化を行った．ここでは，開発フェーズや運用フェーズにて発生し得る変更の取り込みを最小限のワークロードで実施できるようにするために，最適化プロファイルを構成する要素を SQL 一覧（パラメーターの一覧），SQL，最適化プロファイルひな型（目的とするアクセス・プランの骨組みを記述）の 3 要素に分割し，人間による修正が必要な部分を SQL 一覧に集約するという設計を行った．

その結果，図５に示すように，最適化プロファイルひな型に残されているのは定型的なフォーマットの枠組みと，指定するアクセス・プランの骨組みのみとなる．ここで，アクセス･プランの骨組みとは，索引スキャンの指定と，ネスト・ループ結合など，結合の方式や内部表／外部表の結合順序などを指す．これらの骨組みは，性能を最も大きく左右する重要な要素であるが，頻繁な変更が発生することはない．この最適化プロファイルを使用した運用のプロシージャーでは，簡易な SQL 一覧中のパラメーターをセットし，ひな型に対して EXPLAIN 表から取得した SQL を埋め込むことだけで，データベースに登録可能な最適化プロファイルの定義ファイルを生成する．例えば，アプリケーションの開発や保守のフェーズで SQL の修正が行われたとしても，2 表の結合を行う SQL から結合がなくなるなどの，あまりにも極端な修正でない限り，新たに取得した EXPLAIN 結果から取り出した SQL の取り込みを行うのみで，新しい SQL に対応した最適化プロファイルの生成と登録を行うことが可能となっている．

このように，変動要素となり得る SQL や表名，索引名を SQL 一覧へ抽出することで，最適化プロファイルのひな型から変動する要素を除外することを可能とした．このユニークな自動化の設計により，実運用に乗せるための人手による作業や誤りの介在する要素を最小化し，前述の課題 1，課題 2 を解決した．

このプロシージャーによって，XML ファイルの作成に関連する作業の大部分は不要となり，開発中に SQL が変更された場合の取り込みも迅速に行うことが可能となる．ただし，当プロシージャーを採用したとしても，「最適なアクセス・プランが何か」を検討し，それを最適化プロファイルのひな型として作成する部分は，人間による判断に依存しなければならない．また，現時点での最適化プロファイルでは，SQL の全文を組み込んだ定義ファイルを事前にデータベースに登録する必要がある，そのため，パラメーター・マーカーやホスト変数を使用せず，リテラルで記述した条件値を毎回変えて実行するような動的 SQL に対しては，実質的に使用できない．

筆者らは，当プロシージャーを使用することで，500 を超える SQL に対して，意図したアクセス・プランに半自

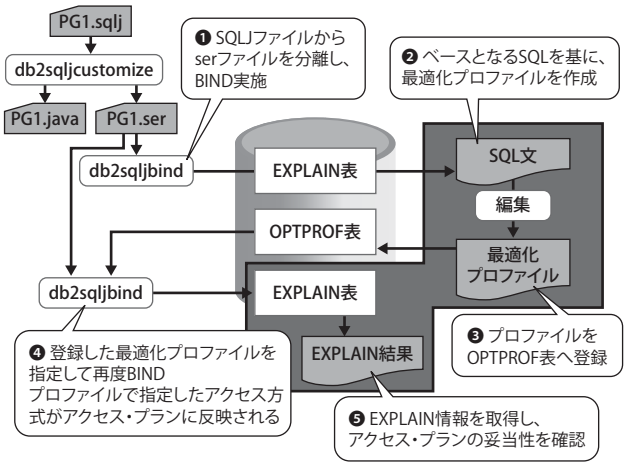

図3. 最適化プロファイルの適用フロー

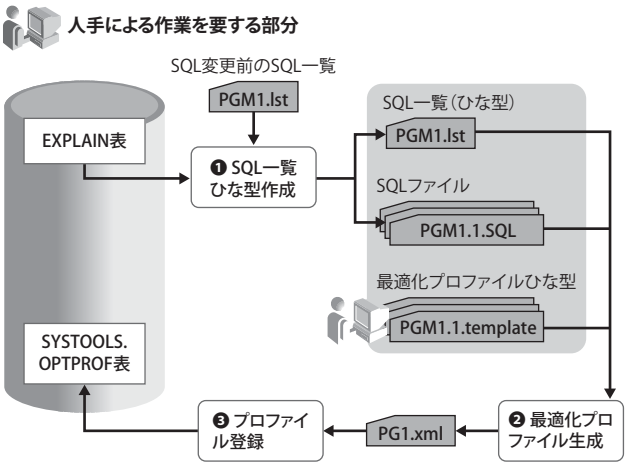

図4. 最適化プロファイル生成のフロー

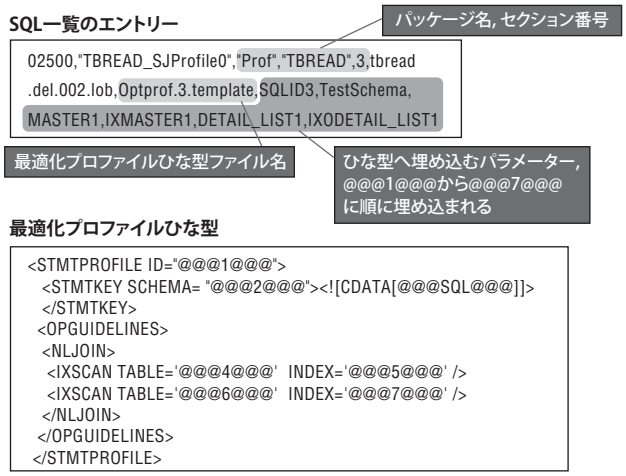

図5. SQL一覧と最適化プロファイルひな型

動的に固定することに成功した．そのシステムでは，同じ類型の表であっても，お客様の事業計画次第でデータの増加率が大きく異なった．そのため，データ量の多寡によってアクセス・プランが大きく異なる例が見られたが，これらの変動要素を最適化プロファイルにより抑止し，5年後のデータ量を見据えた最適なアクセス・プランにそろえることも，あるいは，アクセス・プランの骨組みを定義する最適化プロファイルひな型を変更することで，一括して自由に変更することも可能となった．

SQL 一覧のエントリー例および最適化プロファイルのひな型例を図 5 に示す．この SQL 一覧とひな型を基に生成した最適化プロファイルは，図 1 に示したものと同様となる．SQL 一覧にはパッケージ名称やセクション番号，最適化プロファイルひな型や SQL ファイルのパスが記載され，最適化プロファイルを構成する要素をひも付けている．また，エントリーの末尾にはひな型に入力するパラメーターを集約している．

3. アクセス・プランの同一性チェック

3.1 アクセス・プラン・チェックの必要性

SQL の処理性能を安定化させるためには，アクセス・プランを確認できることが非常に重要である．筆者らの担当したプロジェクトでは，最適化プロファイル適用後に，実際に意図したアクセス・プランとなっているかどうかをチェックする工程を設けた．

しかし，SQL の数が多いシステムでは，チェックのためのワークロードが無視できない．さらに，このチェックは開発時点のみならず，本番リリース時や本番運用後のアプリケーションの修正時にも実施する必要があるため，本番システムの計画停止時間を最小化するためには，可能な限りの自動化を行うことが必須である．

「意図したアクセス・プランが採用されていること」を確認するためには，あるべきアクセス・プランを用意した上で，EXPLAIN 結果を基に何らかのマッチング処理を行って判定する必要がある．DB2/LUW の EXPLAIN 情報は，多くの情報やグラフを含むテキストで出力され，軽微な出力の揺らぎにより不一致となってしまうため，直接の比較に用いるには適さない．

その点を回避するため，筆者らは db2exfmt からの出力情報を抽象化し，アクセス・プランの骨格のみを抽出する独自の手法を考案した．アクセス・プランの管理や分析に関する検討や研究はこれまでにも多数行われてきたが，多数の SQL に対して最小の労力でアクセス・プランの変動を検知する仕組みに関する検討は少ない．本論文で用いた手法は，短時間で多数のアクセス・プランをチェックする際に高い生産性を有する．

3.2 アクセス・プランの抽象化

EXPLAIN 情報には，物理オブジェクト（表や索引）へのアクセスで始まり，結果のアプリケーションへの返却で完了する一連の処理が，処理順序の情報と共に出力されている．この処理順序は「処理 1 への入力となるのは処理 2 と処理 3 である」という形で提供される．この処理単位をオペレーターと呼ぶ．図 6 のように，オペレーターおよびその番号，またそのオペレーターに対する入力を表す“Input Stream”を一固まりにして取り出すことが可能である．つまり，SQL でアクセス対象となる物理オブジェクトから，最終的な RETURN までの各オペレーターでの操作内容とその順序を抽出することが可能である．

第 1 ステップとして，EXPLAIN 情報から，オペレーター番号とそのオペレーターでの操作内容（JOIN や IXSCAN など）および，そのオペレーターへの Input

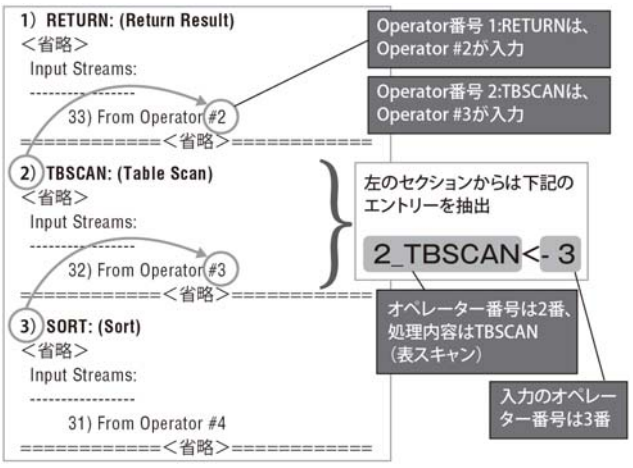

図6. EXPLAIN情報より抜粋

Streamとなるオペレーターの番号を1セットの情報として抽出する．図6に示したのが抽出元の情報であり，図7上側に示したのが抽出結果である．1行目の「1_RETURN <- 2」を例に取ると，先頭の「1」がこのオペレーター番号を表し，次の「RETURN」はオペレーターでの操作内容を，末尾の「2」はInput Streamとなるオペレーターの番号を示す．2行目からオペレーター番号2番の操作は「NLJOIN」であるため，「NLJOIN -> RETURN」という操作の流れがSQL処理中に存在することが読み取れる．

第2ステップとして，図7上側のデータを基に，図7下側の流れ図を生成する．この流れ図には，表や索引へのアクセスから始まって，最終的にデータのRETURNへ至る処理の流れが抽出されている．アクセス・プランの骨格を抽出した情報として，あらかじめ決定しておいた「モデル・アクセス・プラン」との比較に安定して使用できる．参考として，図7での流れ図に対応するグラフを図8に示す．

EXPLAIN情報からの抽出結果（オペレーターと入力番号）

```
1_RETURN <- 2
2_NLJOIN <- 3
2_NLJOIN <- 5
3_FETCH <- 4
3_FETCH <- TUKIV9.MASTER1
4_IXSCAN <- TUKIV9.IX1MASTER1
5_FETCH <- 6
5_FETCH <- TUKIV9.DETAIL_LIST1
6_IXSCAN <- TUKIV9.IX0DETAIL_LIST1
```

上記をもとにオペレーター情報の流れ図を生成

```
1_RETURN <- 2_NLJOIN <- 3_FETCH <- TUKIV9.MASTER1
1_RETURN <- 2_NLJOIN <- 3_FETCH <- 4_IXSCAN <- TUKIV9.IX1MASTER1
1_RETURN <- 2_NLJOIN <- 5_FETCH <- TUKIV9.DETAIL_LIST1
1_RETURN <- 2_NLJOIN <- 5_FETCH <- 6_IXSCAN <-TUKIV9.IX0DETAIL_LIST1
```

図7. EXPLAIN情報から抜粋された流れ図

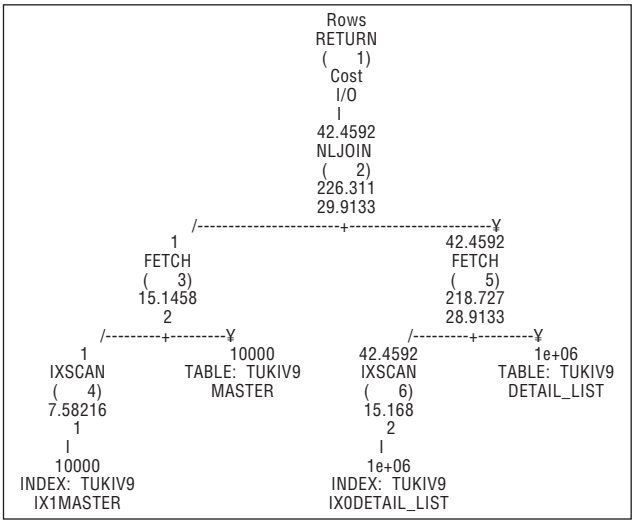

図8.「図7」で示した流れ図の基となったアクセス・プランのグラフ

3.3 モデルとの比較ロジック

2章で述べたように，最適化プロファイルを適用されたSQLであっても，指示した部分以外のアクセス・プランが変動することはあり得る．そのため，モデルとの比較ロジックではアクセス・プランの「揺らぎ」が発生し得ることを前提に，最適化プロファイルで指定された，パフォーマンスのキーとなる部分が正しく適用されていることをチェックする必要がある．筆者らが考案した仕組みでは，EXPLAIN情報から生成した流れ図をそのままモデル・アクセス・プランと比較するのではなく，モデルをより簡素化できるような工夫を行った．

図9に3種類のモデル・アクセス・プランを示した．本プロジェクトで用いた比較ロジックでは，3つのうちいずれを用いても図7下側の流れ図と一致する．それぞれのモデル・アクセス・プランの例で異なるのは，3.1で述べたアクセス・プランの「揺らぎ」への許容度である．

図9の「1」のモデルは，ベースとなった流れ図をそのままチェック用モデルとして使用する例である．この場合，完全に一致するアクセス・プランのみが許容され，2.1末尾で述べた，ソートが追加されたアクセス・プランでは不一致となる．つまり，ほとんどの揺らぎを許容しない．

それに対して「2」のモデルでは，各オペレーター先頭のオペレーター番号を除去している．このようなモデルでは，オペレーター番号の一致チェックは行わず，モデルに存在しないオペレーターが出現した場合も不一致とはしない．つまり，DB2のオプティマイザーが必要に応じて挿入する操作を「揺らぎ」として許容することになる．アクセス・プランの「揺らぎ」が発生する場合の例として，前述したソートのほかに，索引スキャン後にRID（Record ID）でソートした後で表にアクセスするRIDSCANの追加などが挙げられる．

さらに「3」のモデルでは，モデルから結合方式の指

1. ベースとなった流れ図と同様

```
1_RETURN <- 2_NLJOIN <- 3_FETCH <- TUKIV9.MASTER1
1_RETURN <- 2_NLJOIN <- 3_FETCH <- 4_IXSCAN <- TUKIV9.IX1MASTER1
1_RETURN <- 2_NLJOIN <- 5_FETCH <- TUKIV9.DETAIL_LIST1
1_RETURN <- 2_NLJOIN <- 5_FETCH <- 6_IXSCAN <- TUKIV9.IX0DETAIL_LIST1
```

2. オペレーター番号を除外し,番号での不一致を除外

```
RETURN <- NLJOIN <- FETCH <- TUKIV9.MASTER1
RETURN <- NLJOIN <- FETCH <- IXSCAN <- TUKIV9.IX1MASTER1
RETURN <- NLJOIN <- FETCH <- TUKIV9.DETAIL_LIST1
RETURN <- NLJOIN <- FETCH <- IXSCAN <-TUKIV9.IX0DETAIL_LIST1
```

3.「2_NLJOIN」より左を削除し,JOIN方式のチェックを除外

```
FETCH <- TUKIV9.MASTER1
FETCH <- IXSCAN <- TUKIV9.IX1MASTER1
FETCH <- TUKIV9.DETAIL_LIST1
FETCH <- IXSCAN <-TUKIV9.IX0DETAIL_LIST1
```

図9. アクセス・プラン チェック用モデル例

定（NLJOIN）を除去することで，チェックの内容を「指定した索引を使用していること」のみに限定し，結合方式についてはDB2のオプティマイザーによる選択に委ねていることになる．

4. おわりに

DB2/LUWは，より大規模で，よりミッション・クリティカルな領域に使用されながらも，新機能の導入によってアクセス・プランの固定化の運用が一層困難になりつつある．本論文は，ここに，最新技術である最適化プロファイルを利用し，大規模環境の実運用に適用するための3つの大きな課題を，新たに創出した2つの手法を実用化することで，アクセス・プランの意図しない変動を抑止することが可能となることを示した．

その1つ目の手法とは，変動する要素をパラメーター一覧に抽出した上で最適化プロファイルを自動生成するという新しい手法であり，これによって，課題1「SQLと最適化プロファイル間での厳密な同期の必要性」および課題2「最適化プロファイルの汎用的利用が困難」を解決した．

2つ目の手法は，膨大な量のアクセス・プランのチェックにおいて，EXPLAIN情報からアクセス・プランの骨格になる要素のみを取り出して利用し，軽微な変動に影響されないようにする独自の手法である．この手法を使用し，課題3「アクセス・プラン妥当性チェックの困難さ」を解消した．この手法によるアクセス・プラン確認ロジックは，最適化プロファイルには依存せず，単独で使用することも可能である．そのため，「最適化プロファイルの適用までは必要ないが，アクセス・プランの変動確認は行いたい」といった要望にも応えることができ，アクセス・プラン確認手法として高い汎用性を持っている．

筆者らは，これらの2つの手法を，実際に担当した大規模データベースの構築プロジェクトに適用することで，お客様から高い評価をいただいた．

本論文では記載できなかった最適化プロファイルの使用手順や，細かなノウハウについては別途作成した使用ガイド［6］に記載しているため,そちらを参考にされたい．

最適化プロファイルは，本格的な利用に耐えるだけの情報が最近まで整備されていなかったこともあり，実績面からも機能面からもまだまだ発展の余地がある．特に機能面からは，筆者らが案出した手法が，今後の製品機能として実装されていくことが望ましく，DB2/LUWの今後のバージョンでの洗練が期待される．

謝辞

本論文の執筆にあたってさまざまな助言をいただいたプロジェクト・メンバーの方々と，最適化プロファイルの実システムへの適用に向けて叱咤激励をいただいたお客様に，心より感謝いたします．

参考文献

［1］DB2 UDB アクセス・プラン速習，DeveloperWorks Japan，http://www.ibm.com/developerworks/jp/data/library/dataserver/accessplan/
［2］データウェアハウス（DWH）設計ガイド，DeveloperWorks Japan，http://www.ibm.com/developerworks/jp/data/library/100.html#N102C5（2009）．
［3］DB2 Version 9.1 for z/OS，パフォーマンス・モニターおよびチューニング・ガイド，SC88-4790-01（英語原典：SC18-9851-04），pp 284-292（2008）．
［4］DB2 Version9.5 for DB2 on Linux，UNIX，and Windows，データベースの基本，データベース・パフォーマンスのチューニング，SC23-5867-01（英語原典：SC23-5867-01），pp.361-363，pp.420-461（2008）．
［5］Kenneth K. Chen：Influence query optimization with optimization profiles and statistical views in DB2 9，DeveloperWorks，（2006）．
［6］大月 真史：アクセスプランを固定するための最適化プロファイル使用ガイド，DeveloperWorks Japan，http://www.ibm.com/developerworks/jp/data/library/dataserver/j-d_db2optimization/（2009）．
［7］DB2 Version9.5 for DB2 on Linux，UNIX，and Windows，コマンド・リファレンス，SC88-4432-01（英語原典：SC23-5846-01），pp.652-654（2008）．

日本アイ・ビー・エム
システムズ・エンジニアリング株式会社
ソフトウェア・テクノロジー・センター
インフォメーション・マネジメント
主任ITスペシャリスト

大月 真史 Masafumi Otsuki

［プロフィール］
2000年日本IBM入社．金融系のお客様を中心としたプロジェクト参画後，2006年より日本IBMシステムズ・エンジニアリングに異動．ITスペシャリストとして，DB2 for LUWのテクニカル・サポートに従事し，さまざまなプロジェクトへの技術支援を行う．その一環として，本論文で紹介した大規模データベースの構築に携わる．

日本アイ・ビー・エム株式会社
GBS事業，
エンタープライズ・インテグレーション，
システム・アーキテクチャ
エグゼクティブITスペシャリスト

下総 哲郎 Tetsuroh Shimoosa

［プロフィール］
1987年日本IBM入社．インダストリアル・サービスにおける分散システム設計やプロジェクト・リーダーを担当．現場のデータベース・スペシャリストとして，日本IBMのデータベース関する技術コミュニティーをリードし，現在部門全体の技術相談の仕組みをリードしながら，オープン系の大規模データベース設計／構築を中心とするサービス・デリバリーに従事している．