Oracle から MySQL への移行

ストアドプロシージャ、パッケージ、トリガ、スクリプトおよびアプリケーション

ホワイトペーパー

March 2009, Ispirer Systems Ltd.

日本語訳：株式会社ハイ・アベイラビリティ・システムズ

イントロダクション

このホワイトペーパーの目的はOracleからMySQLへのデータベースおよびアプリケーションの移行に影響する要因について説明することです。コストとリスク要因を詳細に説明し、より高い品質の変換を実現するのに役立つツールと方法論についても述べます。

Sun MySQLデータベースは、低いライセンス料金と安価なハードウェアと管理コストにより、企業のデータベースの総所有コスト（TCO）を劇的に下げるのは間違いありません。MYSQLプラットフォームへの移行における最大のリスクは、Oracle上のビジネスロジックの移行リスクと作業の複雑さにあります。特に、既存のアプリケーションがPL/SQLプロシージャ、トリガ、パッケージ、そしてOracle特有のSQL文を多用している場合が問題となります。

OracleからMySQLへの移行は面倒で時間と費用がかかる作業です。しかしながら、実証済みの方法論とツールを利用することにより、要するコストと時間を削減しリスクを大幅に軽減することができます。データベース移行製品SQLWaysを使うことにより、移行作業のアセスメント、計画、そして適切な作業の自動化を行うことができます。自動化ツールの適切な利用と強力なプロジェクト管理の下で、企業は従来の手作業での移行に比較し70%以上のコスト削減を得る事ができます。MySQLの導入によるコストの節約と相まって、ツールによる移行作業の自動化はとても魅力的な選択肢となります。

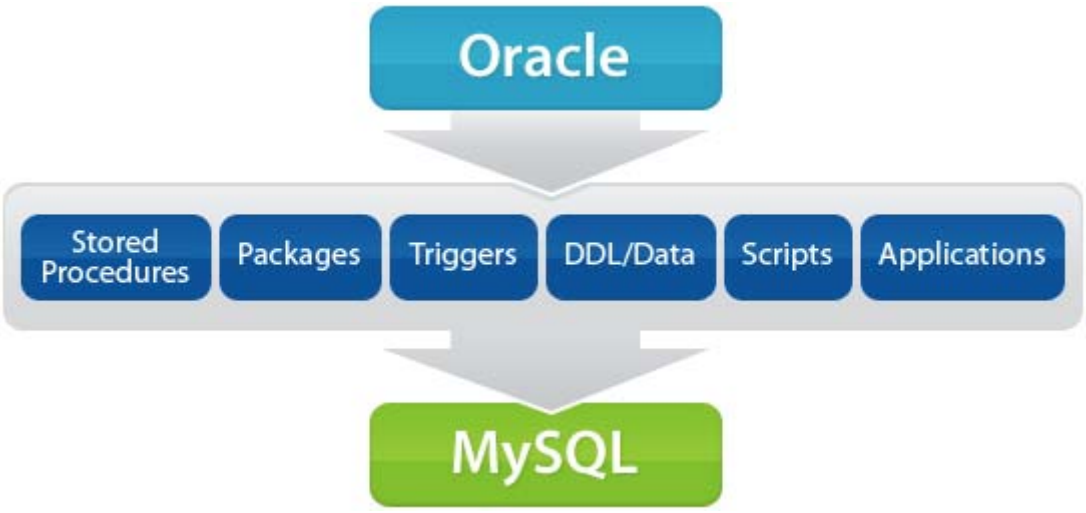
課題

OracleデータベースはPL/SQLストアドプロシージャ、ファンクション、パッケージ、トリガを用いデータベース内部に配置されるアプリケーションロジックを開発できる高度な機能を提供しています。

Oracle PL/SQLは使い易くかつ強力なSQLの手続き型拡張であり、Oracleはパフォーマンスの理由からその使用を強く推奨しています。ほとんどのアプリケーションはPL/SQLを使っており、とても多くの数のプロシージャ、パッケージ、トリガが作成されています。MySQLには類似の機能がありますが、MySQLでPL/SQLそのものを使用することはできません。

PL/SQLは、Oracle独自の構文に加え、Oracle独自機能を含む多くのANSI非互換の機能を提供しています。Oracle独自機能の例を以下に示します：

- パッケージ – 共有パッケージ変数、組込パッケージ
- %TYPE, %ROWTYPE, 例外処理
- オブジェクト指向関連機能：オブジェクトタイプ、ファンクション、コレクション
- ビジネスインテリジェンスおよびXML関連機能など



OracleからMySQLへの移行は簡単な作業ではありませんが、特に上で示したOracle独自機能が使われている場合はさらに難易度が増します。

しかしながら、ターゲットデータベースのテーブル数が比較的少なくビジネスロジックが単純である場合、移行は比較的容易でリスクの低い作業になるでしょう。コストとリスクはプロジェクト毎に異なるため、プロジェクト初期段階でのアセスメントが重要となります。

アセスメント

アセスメントの目的は、Oracleデータベースから MySQLベースのデータベース実装への移行に関する、作業範囲、実現可能性、そしてコストとリスクを明確化する事です。

データベースのアセスメント

最初に、移行対象のデータベースオブジェクトの種類とその数を明らかにしなければなりません。データベースオブジェクトとは以下の様なものです：

- テーブル
- ビュー
- プロシージャ
- ファンクション
- パッケージ
- トリガ
- シーケンス、シノニム等

もしPL/SQLコード（プロシージャ、パッケージ、ファンクション、トリガ）またはOracle独自SQL構文を含むビューやクエリを変換しなければならないのであれば、使用されている機能とその数を調査しなければなりません。調査が必要な項目の例を以下に示します：

- ANSI非互換のSQL関数、演算子、命令文
- 結果セット
- カーソルループ
- 例外処理
- 一時テーブル
- オブジェクトタイプと関数
- コレクション

- 動的 SQL
- 組込パッケージ
- OLAP 機能
- XML 機能　等

調査終了後、発見したOracle独自機能を置き換える事ができるMySQLの同等機能かソリューションを選択できればベストです。代表的なソリューションは後の章に記述されています。

アプリケーションのアセスメント

スキーマやサーバサイドビジネスロジックに加えて、アプリケーションの中のSQL文の修正も必要となるかも知れません。この作業がどれだけかかるのかを見積る事は、移行を完了させるために非常に重要です。

まず始めに、アプリケーションがどの様なAPIを使ってOracleデータベースにアクセスしているのかを調べます。また、Oracle独自コードを含みMySQL向けの修正が必要なソースファイルがどれだけあるのかチェックしておくことも重要です。

多くのアプリケーションはODBC、JDBC、ADO.NETの様な一般的なAPIを使用してOracleにアクセスします。しかし中にはOracle OCIやPro*C/C++のようなOracle固有のAPIを使っているアプリケーションもあるかもしれません。これらの情報を全て洗い出すのは欠かせない作業です。

もしODBC/JDBCドライバ等で標準的なAPIを使用していたとしても、既存のSQL文に大幅な修正が必要になる場合があります。例えば、DECODE関数や古い左辺外部結合構文 (+) は書き換える必要があります。このような固有のSQL文がいくつあるかを見積る事をお奨めします。

もしアプリケーションがOracle OCIのような独自APIを使用しているのであれば、MySQL APIもしくはODBCを使う様にデータベースアクセスのコードを完全に再設計しなければなりません。

アセスメントツール

データベース独自機能がどれだけ使用されているかを把握するのは重要です。「使用されている機能」の見積り作業はどのように行うのが最善でしょうか？

まず以下のテーブルにある様に、テーブル、プロシージャ、ビュー等がいくつあるかを数える事から始めましょう。より詳細な分析を行う為に、IspirerのSQLWays製品を使い全般的な統計情報を収集することができます。

以下はアセスメントの例です：

データベース	個数
Tables	350
Views	280
Procedures	420
Functions	135
Triggers	50
Packages	10
データベース詳細	
BLOBs	37
Outer joins	155
Ref cursors	89
Exceptions	450
Temp tables	34

アプリケーション	
Java/JDBC	590 ファイル
Outer joins	190
SQL functions	356
Result sets	47
移行の手法	自動変換

アセスメントの結果を基に移行計画を立てる事が出来ます。もしプロシージャの数が数十であれば、手作業による変換を検討しても良いでしょう。しかし移行対象のプロシージャが数百や数千になるのであれば市販の自動化移行ツールを検討するべきです。SQLWaysはそのような自動変換機能を提供しています。

コストとリスク

変換プロジェクトにかかるコストと発生するリスクは移行作業の対象範囲に依存します。さらにこれらはデータベースとアプリケーションで使われているOracle特有機能の種類と数にも大きく影響される事に注意しなければなりません。Oracle特有機能が多く使われるほど変換作業は複雑になり、コストがかかります。また、Oracle特有機能が多ければ多いほど自動化ツールの利用が効果を発揮します。

データとDDLの移行コスト

データとDDL（スキーマ）オブジェクトの移行は、それらの作業を支援するツールが市場に多く存在するため、通常はとても容易に実行することができます。

標準的なデータとDDLの移行は以下の変換が含まれます

- データタイプ
- 制約（プライマリ・外部キー、ユニーク制約、ヌル、デフォルト等）
- データ転送
- インデックス

OracleとMySQLのDDL文には異なる構文がありますが、共に類似のデータ型（文字、数値、日付、時刻、LOB）を持っており同様の整合性制約を指定することができます。

DDL/データ移行の工数見積の例（移行ツールを使用した場合）：

データベース	
テーブル数	100 テーブル以下
LOB	10 カラム
テーブルのロー数の合計	10M以下
データサイズ	300 Mb以下
移行作業	
アセスメント	2-8 時間
MySQL の構成	4-16 時間
自動化転送	2-4 時間
試験と構成変更の繰返し	4-12 時間
合計時間	12-40 時間

自動化作業を行うと、DDLとデータの移行コストはテーブルの数とデータ量に比例しなくなります。例えば、テーブル構造やデータ量が同様であれば、100テーブルと300テーブルの移行コストはほとんど変わりません。

テーブルの数やデータ量が増加すると、MySQLデータベースの適切な構成、データ転送の調整、インデックス作成による性能向上の様な作業により多くの時間をかけなければならなくなるでしょう。

標準的なDDLとデータ移行作業のリスクの軽減

標準的なDDL/データ移行は比較的リスクの低い作業でなければなりません。SQLWaysを使うと、評価用にデータベース全体の転送を行ない、データを検査し、アプリケーションを新しいMySQLデータベースで実行してみることができます。

一般的な作業の流れ：

1. データベースのフル転送を評価モードで実行
2. 転送エラーをチェックし、OracleとMySQLのテーブル構造とローの数を比較
3. Oracle SQL*PlusやMySQLクエリブラウザ、コマンドラインユーティリティ等を用いてすべて又は代表的なテーブルのデータを検査
4. MySQLを使用するターゲットアプリケーションを実行しテスト

難易度の高い データ移行

一般的にデータ／DDLの移行はビジネスロジック変換よりも容易な作業ですが、以下のような条件ではデータ／DDL移行作業の複雑さが増加します。

- **大量データ**

もし大量のデータを移行する必要がある場合、MySQLサーバを適切に構成する労力がより多く必要となるでしょう。

大量のデータは特に移行に要する時間に大きく影響します。データ移行にかかる時間を短縮する為に並行処理を行う事が出来ますが、その場合移行作業の複雑さが増加します。

さらに、大量データの転送において、2,3のテーブルでエラーが発生したとしても移行処理全体を再実行することはできないため、エラー処理が複雑になります。

上記の理由によりロー単位にコミットするようなツールは選択できないため、プロジェクトにはデータの一括挿入オプションを持つツールが有用となるでしょう。

- **最小の停止時間**

いくつかのミッションクリティカルな環境においては、サービスの停止時間を最小にする必要があります。この要件を満たすため、データ転送の並行処理や更新のないテーブルを停止時間外で転送する、等の手段を用いて適切な移行処理の設計を行わなければなりません。時には停止時間を短縮するために、データレプリケーションツールを使用する必要があるかも知れません。

- **厳しい性能要件**

環境によってはアプリケーションに対する非常に厳しい性能要件が設定されていることがあります。MySQLへの移行時には、現行と同等の性能を維持するかさらなる性能向上が求められます。このような場合、移行後の性能検証のためのデータベース設計やチューニングに移行作業のテストと同じくらい多くの時間を費やさなければならないでしょう。

難易度の高いDDLとデータ移行作業のリスクの軽減

難しいデータ移行作業は単純なダブルクリックで済ませられるようなものではありません。移行作業のPoC（Proof-of-Concept）を実施して、要件を満たせることを確認する必要があります。

一般的に、複雑なデータ移行プロジェクトでは以下のような移行プロセスが推奨されます。

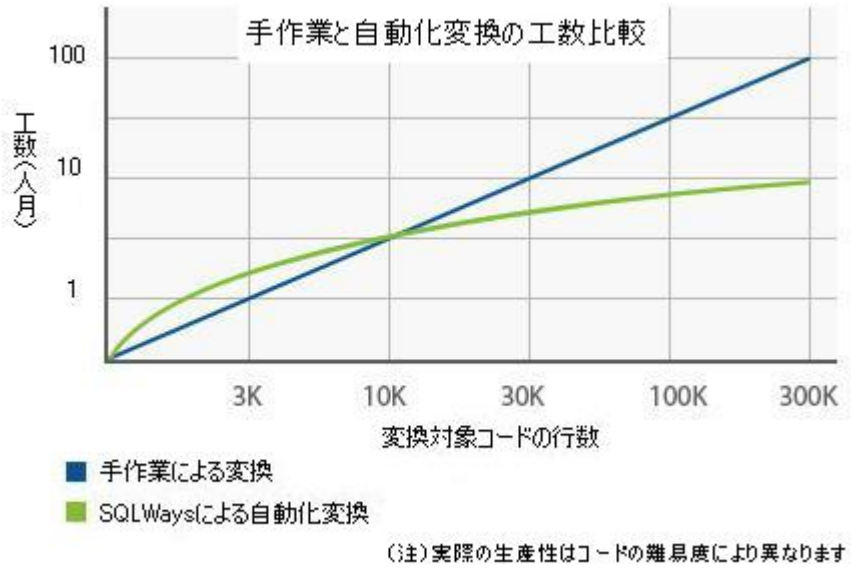
- 要件の実現可能性を確認する為の移行作業PoCの実施
- 本番移行を十分にエミュレートする試験移行と総合的な試験の実施
- 本番移行作業

ビジネスロジック変換のコスト

もし対象のデータベースのプロシージャやトリガの数が数十であれば、手作業でMySQLのストアドプロシージャの構文へ書き直すのは容易な作業です。しかしその数が数千であれば、手作業による変換は大変高くつきます。自動化ツールが役立つかどうかを検討すべきでしょう。

手作業による変換にかかるコストは変換対象のコードの行数に正しく比例します。かたや、自動化ツールはコストの直線的な上昇を抑え、100万行のコードの移行でもほどほどのコストと作業量で行うことができます。

変換対象のコードの行数に依存しますが、SQLWaysのようなツールを使ったビジネスロジックの自動変換は、手作業での変換に比較し1/7から1/10のコストで実現できます。



使われているOracle特有機能の種類と量の多さによって、ビジネスロジックの移行作業の複雑さとツールが提供できる自動化のレベルが決まります。

実際の自動化作業では、SQLWaysのような移行ツールで95%以上のビジネスロジックの変換ができると私たちは考えています。

サーバサイドビジネスロジック移行プロジェクトでの工数見積例：

データベース	
Stored procedures	1000
Triggers	300
Functions	250
Packages	10 (50プロシージャ／パッケージ)
手作業での変換	5,000 時間 （～30 人月）
自動化変換	
アセスメントと ソリューションの検討	48-120 時間
変換の繰返し、分析	120-240 時間
試験	120-240 時間
自動化変換：合計	288-600 時間（2～４人月）

DDL/データ移行とビジネスロジックの移行作業を比較すると、後者が全体のプロジェクト工数の最大95%を占めているのが判るでしょう。これはOracleからMySQLへの大規模な移行プロジェクトで顕著な特徴です。

ビジネスロジック変換におけるリスクの軽減

もし変換対象のコード量が多く、かつOracle特有機能が幅広く使われている場合、変換作業のリスクは極めて大きくなります。そのため幾つかの重要な措置を取りリスクを軽減する必要があります。

- **経験**

移行プロジェクトを担当するスタッフはOracleおよびMySQL両方のデータベースの開発および管理スキルと経験を持っている必要があります。彼らは、移行作業を成功させるために移行作業の対象範囲、課題、役割と段階を明確に理解しておかなければなりません。

- **全般的なアセスメント**

最初の段階で、移行対象のデータベースの全般的なアセスメントを行う必要があります。この結果、変換対象のOracle特有機能や、どのようなソリューションでANSI非準拠のOracle特有機能を置き換えるのかを知る事ができます。

使用されている各機能に対しソリューションが存在するかどうか判断する必要があります。幾つかのOracleの機能はMySQLの同様の機能に容易にマッピングできないため、いくつかの機能の再設計が必要となるかも知れません。

- **コード全量をベースにしたPoC**

SQLWaysのような自動化ツールを使うと、移行性評価の初期段階で容易にコード全体の変換を行う事ができます。これによって潜在的なボトルネックの発見や、ツールの「自動化率」またはプロジェクトへの有用性をより理解することができるため、私たちは全ての複雑な移行プロジェクトにおいてこの作業を行う事をお勧めしています。

より重要なのは、これを行うことにより重いPL/SQLコードの変換が低いコストで可能であると確信できるようになる事です。

- **可能な限り作業を自動化**

手作業による移行は、元々の高いコストに加えて初期段階でボトルネックを発見することが難しいため、後の段階で移行作業の再設計が必要となる事があります。この場合、移行に要する労力とコストがさらに増加する事になります。

比較すると、自動化ツールは変換作業を繰り返し実行する事ができるため、低いコストで高いレベルのフィードバックを得ることができます。これにより手戻りによる多額のコストという罰金を受ける可能性はなくなり、高度に洗練され調整された移行作業を行なう事ができます。

一般的に手作業での変換は退屈な作業であり、非常に人的ミスを起こし易い作業であると言えます。開発者が同じ様なコードについて異なる変換を行ってしまうのは良くある事であり、結果的に試験に多大なコストと時間を要することになります。

- **早期の試験実施**

初期の段階で試験を行う事はプロジェクトのリスクを最小化します。まだアプリケーションレベルの機能試験が出来ない時でも、単体試験またはコードレビューを行うことができます。

プロシージャやファンクションを特定の値で呼び出すテストケースを生成してくれる自動化ツールの機能を利用し、その結果を比較する事が出来ます。この機能はアプリケーションレベルでの機能試験を代替することは出来ませんが、多くの潜在的な問題点の発見に役立ちます。

アプリケーション変換

サーバサイドのビジネスロジック変換に加え、多くの場合アプリケーションもMySQLで動く様に修正する必要があります。

JavaやPowerBuilderアプリケーションの中で、MySQLのSQL構文と異なるANSI 非準拠のSQL文を使っている場合にはそれらを修正する必要があります。

特にOracleからMySQLへの変換で注意すべき最も一般的な構文機能は、Oracleの左側外部結合(+)記法です。DECODE、NVL、SYSDATEの様な関数も全て注意が必要です。

関数の変換は、関数名による単純な検索／置換で行う事はできません。多くの場合、関数のパラメータ構文が異なっていたり、左側外部結合の様にSQL文の修正が必要な事があります。

その上、単純な文字列置換を行うと、文字列やJava言語の命令文など意図しない個所の文字を誤って変更してしまう可能性があります。

より良い手法は、自動的にアプリケーションコードを修正しSQL文をMySQLの構文に適切に変換してくれるSQLWaysのようなツールを使用する事です。

このようなツールはコードの中のSQL文を正しく認識し、変換を行ない、変更を行う毎にレポートを生成します。ツールを利用する事によりアプリケーションの変換作業は大幅に単純化されます。

計画－移行作業の段階

適切な計画を立てる事は移行作業を成功させるためにとても重要です。標準的な移行作業の段階は以下の通りです。

- **アセスメント**

アセスメント（前述）は移行対象のデータベースとアプリケーションを分析し、移行範囲を定義し、MySQLへマッピングする必要があるOracle特有の機能を文書化する目的で行われます。

アセスメントにて収集された情報を基に、今回使用すべき移行作業の手法（手作業、自動化変換）を決定し、移行に関連するコストとリスクの明確化を行う事ができます。

- **初期段階での全量変換－PoC**

もしデータベースに2000個のプロシージャがあるのであれば、PoCの段階にてSQLWaysを実行し、コード全体を変換させる事が出来ます。たとえモジュール単位でのテストと配備を行うつもりであったとしてもこの作業を行う事をお勧めします。

移行作業のごく初期の段階で（自動化ツールを使用する場合）、ツールの実行結果から移行作業に関するおおまかな見通しを得る事が出来ます。これは、手作業による移行がしばしば落とし穴にはまり後戻りを決断するまでに多くの工数を浪費してしまうのとは全く対象的です。

SQLWaysのような自動化ソリューションを用いることで、より統合されかつ均一化された移行手法を取る事が出来ます。手作業での移行のタスクは組織内で分担作業されるため、同じ構文に対し異なる手順や手法が適用されてしまう事がよくあります。結果として作業の成果物の均一化と統合化が進めば進むほど、試験および修正の作業がより容易になります。

理想的には、初期の段階でほぼ100%エラーの無いSQLオブジェクトをMySQLデータベースに生成できる必要があります。これは全てのテーブル、ファンクション、プロシージャ、トリガがMySQLに正しく生成されたということです。

現在入手可能などの変換ツールでも全てのデータベースの100%の変換は非常に困難であるため、Ispirerチームは初期評価期間中にほぼ100%の自動化を達成できるようなツールのカスタマイズサービスを提供しています。

- **実行時、論理、性能試験**

移行作業はしばしばモジュール単位に行われます。サーバサイドビジネスロジックの変換が終了していれば、アプリケーションが未変換でアプリケーションレベルの試験が出来ない場合でも、変換済みのデータベースの試験を行うべきです。

幾つかの代表的なあるいは最も重要なプロシージャを選択してコードレビューを行います。もちろんコードレビューで全ての問題点を洗い出すことはできませんが、初期段階ではこの作業はとても有益です。コードを見ることによりソリューションの適用具合と変換の質を大まかに評価することができます。

より詳細に検証したい機能変換のリストを作成しておくのがベストです。

たとえ変換後のプロシージャやファンクションがデータベースに作成できたとしても、それは重大なエラーを含んでいないということを意味しません。多くのエラーはプロシージャを実行すればすぐに発見することができるため、プロシージャを試験する容易かつ効果的な方法はテストケースを生成する事です。

SQLWaysは異なる入力パラメータによる複数のセットのプロシージャ呼び出しを生成する事ができます。SQLWaysはコードを検査し、どのような値、文字列や日付の制約、制御フロー条件等が使われているかを識別し、多くの条件下で適切なテストケースを生成する事ができます。

より総合的なロジックと性能試験を行うため、実際のデータを使った試験スクリプトを作っていろいろなシナリオを実行することができます。

もしOracleデータベースとアプリケーション向けの自動化された品質保証ソフトウェアを使っているのであれば、それをMySQL向けに更新し総合的な移行試験の確認に使うのを検討するのも良いでしょう。

標準的な変換ソリューションの例

使用される変換のタスクやソリューションは移行プロジェクト毎に異なりますが、それらの多くはあらゆる移行プロジェクトで共通に使われる標準的なものです。

注） 以下に示した全ての変換はSQLWaysが自動的に行ったものです。

DDL

OracleおよびMySQL両方ともCREATE TABLE文をサポートしていますが、構文には多くの違いがあります。

- **データ型**

Oracle	MySQL

CREATE TABLE employees
(
  id **NUMBER(5)**,
  name **VARCHAR2(120)**,
  hire_date **DATE**,
  salary **NUMBER(7)**,
  dept_id **NUMBER(2)**
);
	CREATE TABLE employees
	(
	  id **INT**,
	  name **VARCHAR(120)**,
	  hire_date **DATETIME**,
	  salary **INT**,
	  dept_id **TINYINT**
	);

- **予約語**

OracleとMySQLの予約後は異なっているため、MySQLのクエリに含まれるいくつかのカラム名は引用符で囲まなければなりません。

Oracle	MySQL
SELECT product_id, **limit** **FROM** product_data;	**SELECT** product_id, **`limit`** **FROM** product_data;

クエリと**PL/SQL**コード

主に関数や式の構文変換のためにSQL文を修正する必要があります。PL/SQLはMySQLのSQL手続き構文に完全に変換されなければなりません。

- 外部結合

Oracleは古いアプリケーションで広く使われている特別な外部結合の構文をサポートしています。

Oracle	MySQL
SELECT e.name, d.name **FROM** employees e, departments d **WHERE** e.dept_id = d.id**(+)**;	**SELECT** e.name, d.name **FROM** employees e **LEFT OUTER JOIN** departments d **ON** e.dept_id = d.id;

- カラムへのID値割り当て

Oracleは自動増加カラムをサポートしていないため、アプリケーションやトリガはシーケンスオブジェクトを使用して新しいID値を取得します。

Oracleでは単一のシーケンスオブジェクトを複数のテーブルで使う事ができますが、ほとんどの場合ひとつのテーブルで使用されているため、この機能はMySQLの自動増加カラムに変換する事が出来ます。

自動化変換処理において、SQLWaysはアプリケーションのSQLクエリやINSERT文、プロシージャやトリガを検査し、IDの割り当て処理を確認するとMySQLの自動増加カラムへ変換します。

Oracle	MySQL

CREATE TABLE employees
(
id NUMBER(5) **PRIMARY KEY**,
name VARCHAR2(120),
hire_date DATE,
dept_id NUMBER(2)
);
 	
CREATE TRIGGER emp_id **BEFORE INSERT ON** employees
FOR EACH ROW
BEGIN
SELECT emp_id_seq.nextval
INTO :new.id FROM dual;
END;
	CREATE TABLE employees
	(
	id INT **AUTO_INCREMENT PRIMARY KEY**,
	name VARCHAR(120),
	hire_date DATETIME,
	dept_id TINYINT
	);
	
	-- トリガは不要となります

- 単一イベントでの複数トリガ

Oracleではひとつのテーブルに単一のイベントに対応した複数のトリガを定義する事が出来ます（例えば、employeesテーブルへのINSERTイベントで幾つかのトリガを生成）

MySQLはこれをサポートしていないため、ひとつのイベントに対する全てのコードをひとつのトリガにまとめなければなりません。

- パッケージと共有変数

Oracleでのパッケージは共有変数を持つ関連したプロシージャやファンクションの集まりです。
MySQLではパッケージのプロシージャやファンクションは単体のオブジェクトに変換しなければなりません。

パッケージ変数はひとつのパッケージプロシージャの変数に変更することが出来ます。さらに別のパッケージプロシージャは更新された値を参照または転送することができます。この機能をMySQLで置き換えるには、@記号で始まるセッション変数に変更します。

Oracle	MySQL

CREATE PACKAGE BODY emp_pack
AS
 	
processed NUMBER **DEFAULT 0**;
 	
PROCEDURE new_employee **AS**
BEGIN
...
processed := processed + 1;
END;
 	
PROCEDURE raise_salary **AS**
BEGIN
...
processed := processed + 1;
END;
END;
	CREATE PROCEDURE new_employee
	BEGIN
	...
	IF @processed **IS NULL**
	THEN @processed = 0;
	
	@processed = @processed + 1;
	END;
	
	CREATE PROCEDURE raise_salary
	BEGIN
	...
	
	IF @processed **IS NULL**
	THEN @processed = 0;
	
	@processed := @processed + 1;
	END;
	END;

- **結果セットの返却**

カーソル変数（REF CURSOR）をOUTパラメータとして指定してOracleの結果セットを返す事ができます。多くの場合、これはMySQLの単純なSELECTに変換することができます。

Oracle	MySQL

CREATE PROCEDURE get_salaries
(d_id IN NUMBER, **cur OUT SYS_REFCURSOR**)
AS
BEGIN
 	
OPEN cur FOR
SELECT id, name, salary
FROM employees
WHERE dept_id = d_id
ORDER BY name;
 	
END;
	CREATE PROCEDURE get_salaries (IN d_id INT)
	BEGIN
	
	SELECT id, name, salary
	FROM employees
	WHERE dept_id = d_id
	ORDER BY name;
	
	END;

- **%TYPE および %ROWTYPE データ型定義**

Oracleの%TYPE属性はテーブルのカラムの型を基にしたPL/SQL変数のデータ型を定義するのに使われます。MySQLではデータ型を明示的に指定しなければなりません。

同様のやり方で、%ROWTYPE属性はテーブルのローを基にしたレコード変数を作成するのに使われます。MySQLでは単独の変数を作成し明示的にデータ型を指定しなければなりません。

Oracle	MySQL
v_emp_name employees.name**%TYPE**; v_emp_rec employees**%ROWTYPE**;	v_emp_name **VARCHAR(120)** v_ emp_id **INT** v_ emp_name **VARCHAR(120)** v_ emp_hire_date **DATETIME** v_ emp_salary **INT** v_ emp_dept_id **TINYINT**

- **Java アプリケーションでのSQL変換**

Javaアプリケーションでは、SQL文の構文変換が必要となるでしょう。.

Oracle	MySQL

…
PreparedStatement ps = null;
ResultSet rs = null;
String sql = “SELECT e.name, d.name” +
“FROM employees e, departments d” +
“WHERE e.dept_id = d.id**(+)**”;
 	
ps = conn.prepareStatement(sql);
rs = ps.executeQuery();
…
	…
	PreparedStatement ps = null;
	ResultSet rs = null;
	String sql = “SELECT e.name, d.name” +
	“FROM employees e **LEFT OUTER JOIN**” +
	“departments d **ON** e.dept_id = d.id”;
	
	ps = conn.prepareStatement(sql);
	rs = ps.executeQuery();
	…

- **PowerBuilder アプリケーションでのSQL変換**

PowerBuilderアプリケーションの場合も、SQL文の構文変換が必要となるでしょう。

Oracle	MySQL
datawindow(units=0 processing=0 print.orientation = 0 … print.preview.buttons=no) table(column=(type=char(120) updatewhereclause=yes name=e_name dbname="employees.name") column=(type=char(120) updatewhereclause=yes name=d_name dbname="departments.name") **retrieve**="SELECT e.name, d.name FROM employees e, departments d WHERE e.dept_id = d.id**(+)**”)	datawindow(units=0 processing=0 print.orientation = 0 … print.preview.buttons=no) table(column=(type=char(120) updatewhereclause=yes name=e_name dbname="employees.name") column=(type=char(120) updatewhereclause=yes name=d_name dbname="departments.name") **retrieve**=" SELECT e.name, d.name FROM employees e **LEFT OUTER JOIN** departments d **ON** e.dept_id = d.id”)

非サポート機能に対するワークアラウンドの提供

Oracle PL/SQLの多くの機能は現在のMySQLのSQL手続き型言語ではサポートされていません。もしこのような機能がソースデータベースで使用されている場合、MySQLで同じ動作をさせる為にいろいろなワークアラウンドを使用する必要があります。以下に幾つかの例を示します。

- **PL/SQL コレクション**

一時テーブルとSQL DML操作（SELECT, INSERT, UPDATE, DELETE）を用い、MySQLでこの機能を代替することができます。

- **RAISE_APPLICATION_ERROR**

MySQLのストアドプロシージャで例外を発生させるUDF（ユーザ定義関数）を使用します。

- **UTL_FILE 組込パッケージ**

MySQLストアドプロシージャでファイル操作を行うUDFを使用します。

- **複雑なビジネスロジック**

一般的なソリューションとして、複雑なPL/SQLビジネスロジックはJava言語に変換する事ができます。

まとめ

自動化作業によってMySQLのライセンスモデルへ移行することは非常に大きな価値があります。IspirerのSQLWaysを複雑なOracleからMySQLへの移行プロジェクトに使用すると、品質をより高めると共にお金と時間を節約することができます。既存の業務アプリケーションに対し、ビジネスロジックとデータベースの中味の移行を計画する際には、多くの事柄に留意する必要があります。移行プロジェクトの各段階に対し、適切な計画を立て、分析を行い、細心の注意をはらう必要があります。

ビジネスロジック変換を含むOracleからMySQLへの複雑なデータベース移行は困難な作業でありますが、適切なアプローチと移行ツールを使用する事により、低いコストと低いリスクで移行することができます。IspirerのSQLWays製品とプロフェッショナルサービスは、複雑なビジネスロジック変換において極めて高い価値を提供します。