日日是Oracle APEX

Oracle APEXを使った作業をしていて、気の付いたところを忘れないようにメモをとります。

2021年2月13日土曜日

Oracle Database Multilingual Engine (MLE) の記事３選

最近、いくつかOracle Database Multilingual Engine (MLE) の記事が出てきています。

Multilingual Engine: Executing JavaScript in Oracle Database

https://medium.com/graalvm/mle-executing-javascript-in-oracle-database-c545feb1a010

ひとつめはAlina Yurenkoさん、OracleでGraalVMのDeveloper Advocateをしている方による以下の記事です。

前半でDBMS_MLEパッケージを利用したJavaScriptのコードの実行方法について、平易に解説しています。

いくつか説明を抜粋します。

- JavaScriptのコードはPL/SQLの代替引用符(q'~と~'といった書き方)を使って記述すると、文字のエスケープを気にしなくて済みます。
- console.log()はDBMS_OUTPUTと同じです。SERVEROUTPUTの設定の影響も受けます。
- ひとつのデータベースのセッション中で、DBMS_MLE.create_context()を複数回呼び出すことで、複数の独立したコンテキストを生成できます。ただしメモリは消費するので、必要なときだけコンテキストは生成し、不要になったら削除しましょう。SQLカーソルのようなリソースと扱いは同じです。
- SQLとPL/SQLを実行するためにmle-js-oracledbというJavaScriptモジュールを提供しています。
- mle-js-oracledbは一般的なNode.js用のOracle Databaseドライバ(node-oracledb)を模して作られています。
- Oracleの持つNUMBER型については、JavaScriptの浮動小数点型に変換すると値が変わるため、OracleNumberオブジェクトを提供しています。

後半でMultilingual Engineの実装について、簡単に解説しています。

- GraalVM Native Imageにより、MLEのランタイムとGraalVMが提供しているJavaScriptラインタイムを共有ライブラリとして生成し、それをデータベースのプロセスが必要に応じてロードしています。
- Cのコードと、GraalVMのNative Imageのファンクションの間でのやり取りは、とても効率的です。
- MLE Native Imageのロードを短期間に行うため、MLEを構成する多くのJava Classはイメージをビルドする時点で初期化しています。

- 読み出しのみを行う（メモリの）ヒープ領域はイメージのビルド時点で初期化され、データーベースのプロセス間で共用されます。そのため、全体のメモリ使用量が削減されます。
- MLEはOracle Databaseのリソース・マネージャのポリシーによって制御されます。
- MLEは直接OSからメモリを取得することはなく、データベースから取得します。
- ひとつのデータベース・セッション内で複数の(MLEの)コンテキスト(Polyglot Context)が持てますが、これはGraalVM自体に組み込まれているものです。実行エンジン(Polyglot Engine)は共用しているため、メモリの使用量や実行する(同じ)JavaScriptのコードをパースする時間が短縮されます。
- mle-js-oracledbモジュールはTruffle language interoperability protocol経由でMLE SQL Driverを呼び出しています。これにより、２つのJavaScriptのファンクション間での呼び出しと、発生するオーバーヘッドはほぼ変わりません。

MLE and the Future of Server-Side Programming in Oracle APEX

https://blogs.oracle.com/apex/mle-and-the-future-of-server-side-programming-in-oracle-apex

ふたつめは、Oracle APEXの開発チームに所属しているSalim Hlayelさんの記事です。

Oracle Database 21cは、Always FreeのAutonomous Databaseとしていくつかのリージョンで先行利用することができます。

- Germany Central (Frankfurt)
- US East (Ashburn)
- US West (Phoenix)
- UK South (London)

この中のAshburnで21cのAlways FreeのADBを作成し、Oracle APEXのデモでよく利用されているプロジェクトとタスクのアプリケーションにJavaScriptで記述したプロセスを追加しています。

MLEが利用可能なデータベースの作成から、Oracle APEXのアプリケーションへのの適用まで、ステップバイステップで説明しています。これからOracle APEXでサーバー側にJavaScriptでコードを書こうと検討されている方とって、とても役に立つ記事です。

JavaScript as a Server-Side Language in Oracle APEX 20.2

https://medium.com/graalvm/javascript-as-a-server-side-language-in-oracle-apex-20-2-457e073ca4ca

みっつめは、Oracle APEXの開発チームに最近加わったStefan Dobreさんによる記事です。

サーバー側でJavaScriptによるコーディングを行う利点を考察しています。

- 単にSQLを実行するだけなら、PL/SQLの方が簡潔に書けるでしょう。
- 正規表現についてはJavaScriptの方が互換性の心配がない。
- JSONの扱いはJavaScriptの方が容易です。

また、彼によるとMLEのデータベース・インターフェースは(公式なAPIのドキュメンテーションはまだリリースされていない)Node.js Oracle Driverのサブセットだが、ファンクションが非同期から同期に変更されていて、そこが大きな違い、とのことです。

Oracle APEXでJavaScriptのコードを記述する際には、mle-js-bindingsといったモジュールを明示的にrequireする必要がありません。なので、どのようにMLEのコンテキストを初期化しているか説明しています。Oracle APEXのページの処理が行われる際に、色々な場所でJavaScriptのコードを記述できますが、それを実行する際にMLEのコンテキストが生成されます。そして、すでにMLEのコンテキストが生成されている場合は、それを再利用します。結果として、ページの処理１回あたり、最大１つのMLEコンテキストが生成されることになります。

MLEのコンテキスト(Oracle APEX限定かどうかは分かりませんでした)には、globalThisというオブジェクトが提供されていて、この名前空間にファンクション、オブジェクト、変数などを割り当てることができます。これはブラウザのwindowオブジェクトに似ています。PL/SQLによるプロセス記述にて同様の実装をするには、PL/SQLの場合はページ・アイテムかパッケージ変数をを使う必要があります。

最後にqrcode-generator、validator.jsそれとmarked.jsを読み込んでモジュールとして活用する方法を示しています。これらのJavaScriptのファイルを一旦、CLOBの列を持つ表に取り込んで、それから、その内容をPlolyglot.eval()を使って利用可能にしているようです。

Alinaの記事からは、JavaScriptコードはコンテキスト間では共有されるが、実行エンジンの間では共用されないように読め、また、MLEはデータベースのセッション内で複数のMLEコンテキストを持てるが、Polyglot Engineはセッション単位であると読めます。そのため、これらのJavaScriptのモジュールはOracle APEXの１ページ処理ごとに必ず１回はPolyglot.eval()が呼び出されるのではないかと思われます。

これ以降はqrcode-generator, validator.js、marked.jsを使用した実装を紹介しています。ある程度、JavaScriptにて複雑な処理を記述する場合はモジュールは利用する必要が出てくるでしょう。ですので、サーバー側でJavaScriptを使うことを検討しているのであれば、読んでおくべき記事です。

以上です。ぜひ元記事の方も参照してください。

完

Yuji N. 時刻: 11:38

共有

自己紹介

Yuji N.

日本オラクル株式会社に勤務していて、Oracle APEXのGroundbreaker Advocateを拝命しました。
こちらの記事につきましては、免責事項の参照をお願いいたします。

詳細プロフィールを表示