ORACLE
FUSION MIDDLEWARE
STREAM EXPLORER
PLATFORM

Oracle Stream Explorerの概要

ソフトウェア・コーディング不要の高速データおよびイベント処理

Oracle ホワイト・ペーパー | **2015** 年 **3** 月

ORACLE

目次

はじめに

あなたが起きた瞬間から 1 日が終わるまで、数えきれないほどのイベントが私たちの知らぬ間にいたるところで発生しています。一般的な辞書によると、イベントとは何らかが起きる事象です。たとえば、家の持ち主によるサーモスタットの調整や、食料品店でのクレジット・カード処理、高速道路の料金所の通過などがあります。最終的な分析に含まれるすべての要素は事実上イベントと見なすことができ、多くの種類のイベントはおそらく互いに関連しています。イベントに注意を払うことが重要なのは、今何が起きているかを教えてくれ、自分たちを取り巻く状況やもっと広い世界の現状について気づかせてくれるからです。

イベントの関連性とその結果を分析するテクニックはイベント処理と呼ばれます。イベント処理が扱うのは進行中のイベントであり、高速道路での燃料切れを防ぐために自動車の燃料警告灯に気を配るのと同じように、措置を講じる場合はこれが理想的なタイミングだからです。いったん発生したイベントは過去のものになり、単なる事実の記録になります。過去の事実を分析することにも価値はありますが、発生したイベントやそのイベントが示す状況によっては、失われた機会や見過ごされた脅威を表している場合があります。このため、特に 21 世紀型企業にとって、イベント処理はこれまでになく重要になっています。

ほぼすべてに近い業種で、すでに何らかの形でイベント処理が使用されているもの、とほとんどの人は考えていますが、実際に使用している業界はほんの一部です。大半の企業は EDA（イベント駆動型アーキテクチャ）を使用して異なるシステム間でイベントを交換しており、疎結合やメッセージ・ルーティングといった EDA 固有の特性を使用しているにもかかわらず、実際はイベント処理ではなくイベント配信のみを実行しています。いくつか名前を挙げると、自動取引やオンライン・ゲームがイベント処理を使用している業種の例です。ただし、これらはもともとイベント駆動型の業界であり、イベント処理は基幹ビジネスの一部です。では、イベント処理がそれほど重要であるならば、なぜ残りの業界はこれを利用できていないのでしょうか。

一番の原因は現在のテクノロジーが持つ抽象化レベルにあると考えられます。ビジネス・インテリジェンスと比べると、イベント処理テクノロジーの抽象化レベルは低いものです。OEP（Oracle Event Processing）のようなもっともパワフルなイベント処理テクノロジーでも開発者コミュニティに焦点を合わせています。このため、OEP はイベント処理アプリケーションの作成やテスト、デバッグ、デプロイを行うための包括的なツール・セットを提供しており、待機時間が非常に短くスループットの高い、フォルト・トレラントのような機能含んでいます。しかし、これには代償がつきもので、分析に関心を寄せるユーザーがテクノロジーの技術的内容に触れる事態になっています。

ビジネス・ユーザーを戸惑わせがちな技術情報をすべて理解しなくても、イベント処理を実行できる機能を提供する製品を望む声が、業界には明らかにあります。これこそが、オラクルが新しいOracle Stream Explorer 製品に着手した理由であり、オラクルは直感的でシンプルなユーザー・インタフェースによってイベント処理アプリケーションを作成できるプラットフォームをビジネス・ユーザーに提供することを目指しています。Oracle Stream Explorer は社内使用と Oracle Cloud での SaaS 使用の両方が可能な Web 対応アプリケーションであり、イベント・ストリームのリアルタイム分析に関心のあるユーザーに対してコーディング不要の環境を提供します。

本書の目的は、Oracle Stream Explorer を使用したアプリケーション構築の開始に必要な基本情報を提供することです。おもな製品機能の概要を提供するとともに、サンプル・アプリケーションの開発方法を、興味深い事例に基づいてステップごとに紹介します。

シェイプ、ストリーム、リファレンス、エクスプロレーション

この項では、イベント処理アプリケーションの開発中に作成される、もっとも重要なアーティファクトについて紹介します。Oracle Stream Explorer を初めて使用する場合、製品を使ってみる前にこの項を一読することを強く推奨します。

シェイプ

多くの心理学者によると、人間は何か新しいものを学習し始めるとき、頭の中でその対象をジオンと呼ばれる単純な幾何学形状に分解します。ジオンは円柱やれんが、くさび、円、長方形などの単純な 2 次元または 3 次元の形態をとります。全体として観察した場合、すべてのジオンはより高位のジオン（車など）を表しますが、個々に観察した場合もそれぞれのジオンが何らかを表しています。これは、脳内で、対象物の構造表現に各ジオンが対応しているためです。現実世界では、これらのジオンはシェイプ（形）として知られています。

シェイプはすべての対象物を分類するための基本的構成要素です。すべての対象物はシェイプであり、構造表現を持ちます。イベント処理アプリケーションの構築でもこれに変わりはありません。分析対象データや分析に役立つデータのすべての断片がシェイプになります。データセットの実用的な表現を提供するには、シェイプを作成する必要があります。シェイプを作成せずに Oracle Stream Explorer でデータを処理することはできません。

技術的な観点から見ると、Oracle Stream Explorer におけるシェイプはデータセットに関するメタデータを提供するある種のアーティファクトになります。名前と属性リスト、それぞれのデータ型を持ち、一般にソース・タイプと関連付けられた場合に使用されます。ソース・タイプはデータの取得元によって決まり、各ソース・タイプは内部で OEP アダプタを介して実装されています。シェイプに関するもう 1 つの重要な側面として、シェイプが永続化されることはありません。イベント・ストリームの分析中、データはメモリに読み込まれますが、分析が終了するとすぐにすべてのデータは破棄されます。

ストリームとリファレンス

一般に、イベントはその時間的な状態によって分類できます。過去の事実を表す処理済みデータがあれば、現在起きていることを表す進行中のイベントもあります。イベント処理が現時点で起きて

いることの発見にほかならないのは事実ですが、ほとんどの場合、現在発生しているイベントだけではコンテキスト全体を理解するためのデータが十分ではありません。前述したとおり、過去の事実を分析することには価値がありますが、イベント処理の分野でのこの価値は、過去のイベントを使用して現在のイベント・ストリームに前後関係をもたらすという考えに基づいています。

たとえば、商品を載せた多数の輸送船が船の位置に関するイベントを絶えず送信しており、これらのイベントから特定の船が遅れる（つまり商品の配達が遅れる）タイミングを検出して警告する必要があるとしましょう。イベントには船の識別子が含まれると仮定しても、船の出発地や船籍などを特定するには外部データソースが必要です。この状況において、船から送られるイベントは進行中のイベントであり、現時点で発生中であることを表しています、また、それぞれの船に関する詳細は以前に処理されたデータを表しているため、過去の事実ということになります。

Oracle Stream Explorer では、進行中イのベントを表すために使用される際限のない連続データをストリームと呼びます。ストリームは決して停止することなく持続しており、イベント処理分析に未加工のデータを提供します。しかし、ほとんどの分析ではこの文脈データまたは履歴データの一部が実用的なものである必要があります。ストリームに対して前後関係をさらに追加するために使用されたデータを表す統計データは、リファレンスと呼ばれます。リファレンスは静的データであり、その用途から名前が付けられています。リファレンスは、処理済みデータを参照するために使用されます。時間的な状態の観点から見ると、ストリームは現時点で起きていることを表し、リファレンスは過去の事実を表しています。

エクスプロレーションとパターン

イベント処理アプリケーションの開発では、所定のビジネス目標を達成するために実行する必要のある一連の共通アクティビティがあります。これには、接合や時間的制約の作成、集計の実施、出力結果のフィルタリングとカスタマイズなどが含まれます。ほとんどの場合、これらのアクティビティは求められたビジネス目標に達するまで繰り返し実行する必要があります。Oracle Stream Explorer では、エクスプロレーションと呼ばれるアーティファクトを介してこれを実現します。

すべてのエクスプロレーションはソース・リストに関連付ける必要があり、ソースはストリームまたはリファレンスになります。ただし、ソース・リスト内には少なくとも 1 つのストリームが含まれる必要があり、これはリスト内の最初のソースでなければなりません。残りのソースは別のストリームであるか、またはリファレンスであり、リスト内で使用されるソースの数に制限はありません。ソースおよびエクスプロレーション間の関係から、エクスプロレーション内での使用を意図したすべてのソースは、そのエクスプロレーションよりも先に作成する必要があります。

エクスプロレーションの出力結果は常に新規ストリームになります。このため、いったん公開されたエクスプロレーションは、その他のストリームと同様に、別のエクスプロレーションのソース・リスト内で使用できます。図 1 にソースとエクスプロレーションの関係を示します。

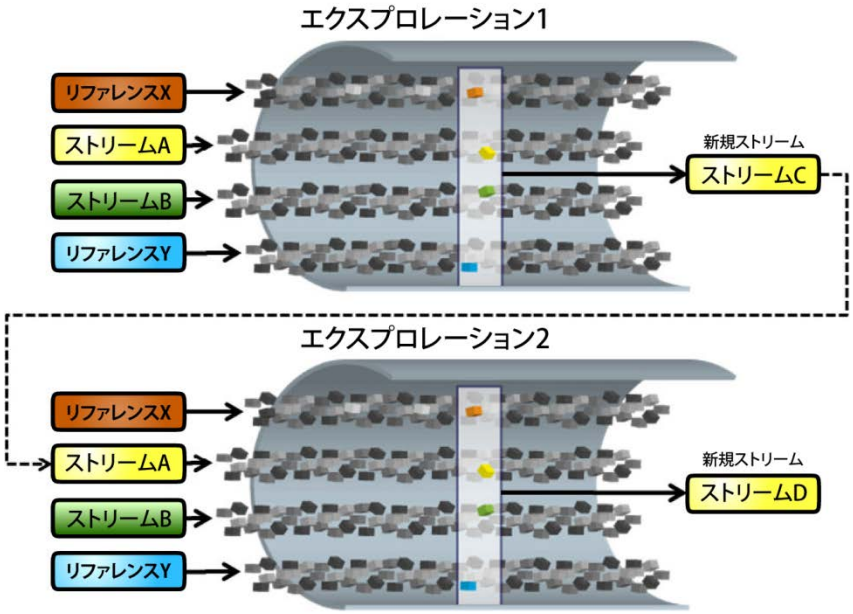

図1.ソースとエクスプロレーションの関係

エクスプロレーションの処理には長い時間がかかる場合があり、特に考察の対象シナリオに複雑なロジックが必要とされる場合はその可能性が高くなります。この理由から、Oracle Stream Explorer は、パターンと呼ばれる事前構築済みのエクスプロレーションを利用できるようにしています。パターンは反復する一般的な問題に対する汎用ソリューションであり、詳細な実装ではなくビジネス上の問題に焦点を合わせ続けるために役立ちます。

パターンの使用法は非常に簡単です。実装する必要のあるパターンを選ぶと具体的なデータまたはキー・フィールドが要求され、すべてを指定した後でエクスプロレーション全体が自動的に生成されます。Oracle Stream Explorer で現在使用できるパターンは次のとおりです。

» Top N：最初の"N"個のイベントをイベント・ストリームから取得するために使用します。
» Bottom N：最後の"N"個のイベントをイベント・ストリームから取得するために使用します。
» Up Trend：数値フィールドで指定した傾向の変化で値の上昇を示す場合に、これを検出します。
» Down Trend：数値フィールドで指定した傾向の変化で値の下降を示す場合に、これを検出します。
» Fluctuation：特定のフィールド値が指定した時間枠内で特定の方向（上方または下方）に変化した場合に、これを検出します。
» Eliminate Duplicates：イベント・ストリーム内の重複イベントを除外します。
» Detect Duplicates：指定した期間内に特定のデータ・フィールドに重複値が含まれる場合に、これを検出します。
» Detect Missing Event：指定した時間枠内に期待したイベントが発生しなかった場合に、これを検出します。
» W：指定した時間枠内に特定のフィールド値が"W"型に上がって下がった場合に、これを検出します。
» Inverse W：逆 W 型を検出するために使用します。

いったん構築したエクスプロレーションは別のエクスプロレーションのソースとして使用するか、またはエクスポートできます。エクスプロレーションをエクスポートするかどうかの判断は通常、

エクスプロレーションが OEP アプリケーションとして使用されることを意図しているどうかによって決まります。つまり、もともとはビジネス目標を達成するために Oracle Stream Explorer で作成されたものが、後から OEP で拡張されて統合やセキュリティ、パフォーマンス、スケーラビリティ、フォルト・トレランスなどの技術要素を扱うことになっている場合にあたります。

前述したとおり、Oracle Stream Explorer の基盤は OEP であり、生成されたエクスプロレーションは OEP アプリケーションとしてデプロイされた EPN（イベント処理ネットワーク）にすぎません。図 2 に EPN の例を示します。エクスプロレーションは EPN であるため、Oracle Stream Explorer からエクスポートして、Oracle JDeveloper などの開発環境に再度インポートすることができます。エクスプロレーションをエクスポートすると、イベント・タイプやアダプタ、キャッシュ、チャネル、プロセッサ、CQL（Continuous Query Language）文などの一般的な EPN アーティファクトを含む JAR ファイルが生成されます。

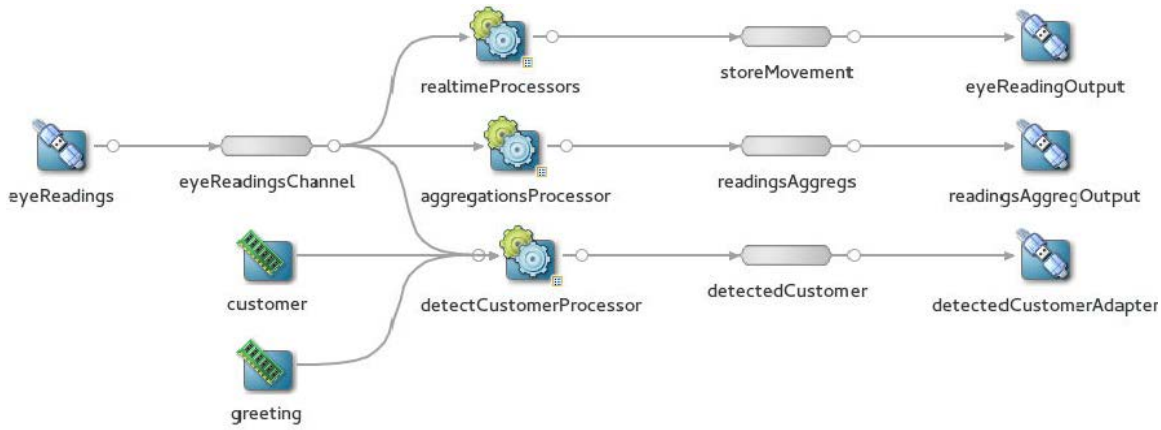

図2.エクスプロレーションの実装から生成されたEPNの例

事例：「マイノリティ・リポート」における買い物シーンの実装

2002 年 7 月、20 世紀フォックス・スタジオは映画「マイノリティ・リポート」を公開しました。Steven Spielberg が監督を、Tom Cruise が主演を務めたこの映画は、あっという間に過去最高の成績を収めた SF 映画の 1 つになりました。この映画では犯罪予防と呼ばれる特別な警察プログラムを中心に物語が展開します。犯罪予防プログラムは、特別な空間に横たわる 3 名の予知能力者が見る未来を予見する映像を使って、殺人が起きる前にこれを阻止するべく、先制攻撃的に警官隊を配備して未来の殺人犯を逮捕するものです。この映画で犯罪予防プログラムが設立されたのは 2050 年のワシントン DC であり、Tom Cruise が犯罪予防プログラムを担当する John Anderton 警部役を演じています。極めて興味深いストーリーに加えて、数多くのシーンでハイテクが使われており、近未来の予想図を描き出しています。

あるシーンで、John Anderton は予知能力者がいる空間に入り込む必要に迫られますが、この時点で John は将来における犯罪の容疑者であったために難題に直面することになります。2050 年、ほとんどの公共の場には全市民を識別できる網膜スキャナが設置されています。逮捕されないようにするため、John は自身の眼球を取り除くほかなく、Yakamoto という死亡した人物の眼球と取り替える手術を行います。手術後、Yakamoto 氏の眼を手に入れた John は奇妙な状況に陥ります。新しい洋服を探して GAP の店舗に入った際の「こんにちは、Yakamoto さん。いつもありがとうございます」という挨拶に驚くことになります。

このシーンが教えてくれるのは、センサー技術とイベント処理を組み合わせることで、個人に合わせたカスタマ・エクスペリエンスの提供が可能になり、顧客の個人的好みに関する情報を使って、必要なときに適切な反応を返すことができるということです。ここで使われるセンサーは網膜スキャナであり、途切れることなく人間の網膜を読み取って、その結果をイベント処理対応テクノロジーで処理するために送信しています。イベント処理部分は必要に応じて作動し、複数のイベントのストリームから店舗に近いイベントを検出し、人物を認識した後にカスタム・メッセージで挨拶します。

対象人物が店舗近くにいる間に、イベント処理システムがカスタム・メッセージを起動する必要がなければ、このシナリオの実現はそれほど困難ではないでしょう。仮に現在のビジネス・インテリジェンス・テクニックを使用した場合、人物のスキャン・イベントはステージング・データベースに格納され、店舗近くにいたときから 1 時間後から 1 日後、場合によっては 1 週間後に、スケジュール設定されたジョブによって最新の"N"個のスキャンがロードされることになります。意味のある処理にするには、人物が店舗近くにいる間にカスタム・メッセージを配信しなければなりませんが、この状態は数秒しか続きません。

これこそがイベント処理テクノロジーの威力が発揮されるケースであり、この種の SF シーンを可能にするシステムの構築が実現可能になります。イベント処理では現在発生している事象の分析が可能であり、引き続き継続中のイベントを処理して、イベント間の複雑な関係を検出することができます。映画「マイノリティ・リポート」の買い物シーンがこの項で開発する事例となります。この事例は、Oracle Stream Explorer でエクスプロレーションを使用して構築します。店舗に近づいた人物を検出し、その人物を認識し、最後にカスタムの挨拶メッセージを表示するまでが対象になります。実装を可能にするため、以下の想定が当てはまるものとします。

» それぞれの網膜スキャン・イベントには人物の網膜と位置が含まれる

» すべての顧客情報はデータベース表から取得される

» カスタム・メッセージはデータベース表から取得される

この事例の開発は 2 つの部分に分けられます。第 1 部でソリューション設計に関する詳細情報を扱い、すべての必要なアーティファクトを特定します。第 2 部の対象は Oracle Stream Explorer でのアーティファクト開発であり、アーティファクトの作成方法と構成内容の処理方法を紹介します。

第1部：シナリオ用ソリューション設計の作成

Oracle Stream Explorer で実装するソリューションを設計する場合、シェイプから始まる必要なアーティファクトすべてを詳しく指定する必要があります。シェイプはイベント処理アプリケーション内を流れる全データを表していることから、ソリューション設計の第一歩は概念モデルの構築になります。すでに使用可能なシェイプとこれから必要になるシェイプの 2 つのカテゴリにシェイプを分けます。図 3 にこの事例の概念モデルを示します。

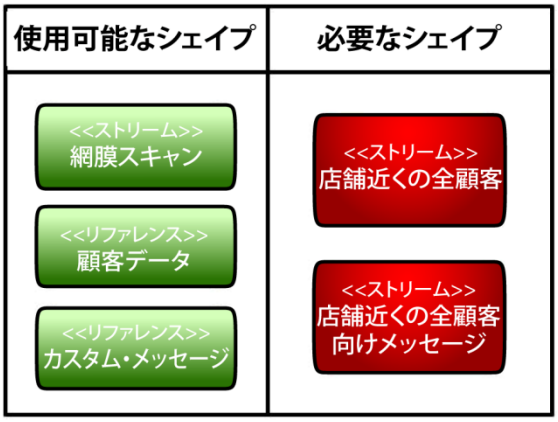

図3.事例の概念モデルに示したシェイプ

用途に従って各シェイプを分類するために、概念モデルに定型化が含まれる場合があります。進行中イベントに関連付けられるシェイプはストリームとして定型化され、すでに処理済みのデータに関連付けられるシェイプはリファレンスとして定型化されます。必要なカテゴリ内に含まれるすべてのシェイプはストリームとしても定型化できます。これは、Oracle Stream Explorer の見方では、シェイプはエクスプロレーションから派生したものであり、エクスプロレーションの出力結果は常に新規ストリームになるためです。このため、ストリームの代わりに定型化したエクスプロレーションを使用しても間違いとは見なされません。最終的な分析ではすべてのエクスプロレーションがストリームになります。

次のステップでは、すべてのシェイプ間にある因果関係を盛り込み、1 つまたは複数のシェイプを処理することが、新しいシェイプの作成にどうつながるかに重点を置きます。ここで処理という用語は、エクスプロレーションの実行中に発生する接合、時間的制約、集計、フィルタリングの各アクティビティを使用することを意味します。また、このステップでの重要な処理として、各シェイプに属性を追加して、使用可能なシェイプと必要になるシェイプ間で属性同士をマッピングします。図 4 に、因果関係を含む概念モデルを示します。

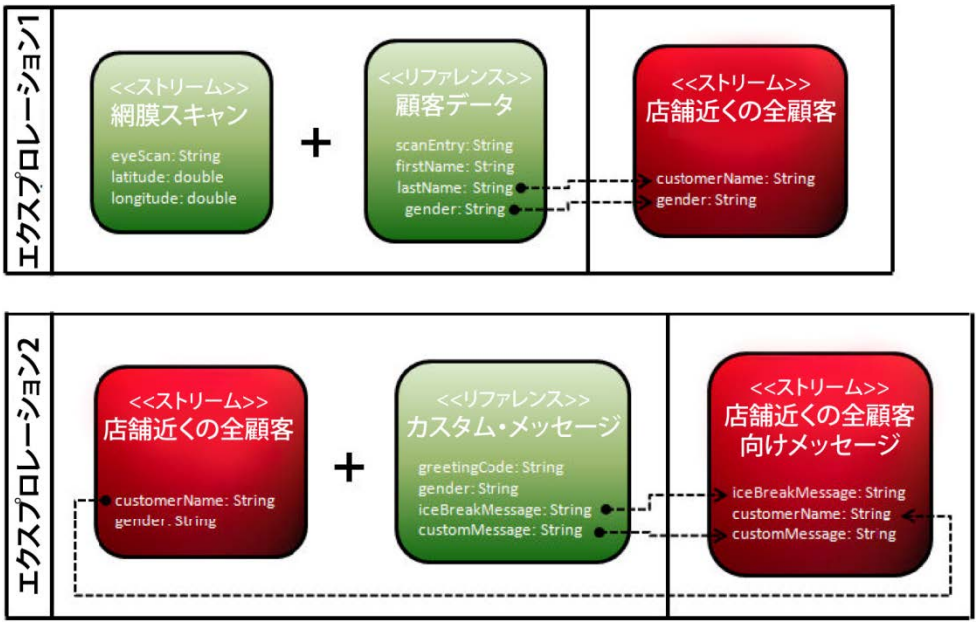

図4.因果関係を含めて改善した概念モデル

図 4 に示した 2 つの枠は、Oracle Stream Explorer で構築する必要のあるエクスプロレーションを表しています。構築する必要のあるエクスプロレーションの数を事前に突き止めることが重要です。これにより、実装面から見てどのくらいの開発作業が必要になるかが明らかになります。異なるエクスプロレーション間での因果関係の実装を義務付けるルールは存在しませんが、ベスト・プラクティスとしてビジネス上の問題をより小さいエクスプロレーションに分割すると、アーティファクトの再利用が推進され、無理のないレベルまで複雑さを緩和できます。

エクスプロレーションには意味のある名前を付ける必要があります。経験から言って、それぞれのエクスプロレーションに、作成する予定のシェイプ名を含めると良いでしょう。エクスプロレーションの実行中にビジネス・ルールを適用する必要がある場合は、ベスト・プラクティスとして、該当するビジネス・ルールの説明を提供することを推奨します。たとえば、1 番目のエクスプロレーションには、顧客が店舗の近くにいるかどうかを検出するため、網膜スキャンのストリームに含まれる latitude 属性と longitude 属性がビジネス・ルールとして含まれています。また、店舗近くにいる人物を特定するために、網膜スキャン・ストリーム内の eyeScan 属性と顧客リファレンス内の scanEntry 属性を関連付ける必要がありますが、これもビジネス・ルールと見なされています。

この説明を利用する対象者は Oracle Stream Explorer でアーティファクトを実装するユーザーであり、概念モデルを構築した人物であることもあれば、別のユーザーであることもあります。Oracle Stream Explorer は細かい実装を高いレベルで抽象化しているため、ほとんどの場合、概念モデルを構築したユーザーがアーティファクトの実装も行います。このため、ソフトウェア開発スキルをほとんど（またはまったく）持たない担当者でも直接製品を使用して、最初から最後まで実装に責任を持つことができます。

異なる担当者が協力してシナリオを実装する場合は、ビジネス・ルールの内容を記述しておくと、プロジェクトの実装全体を通じたコミュニケーションで役立ちます。これは特に、ソリューションの設計者と実装者が異なる会社に属する場合や、地理的に離れた場所にいる場合に重要になります。図 5 に最終版の概念モデルを示します。

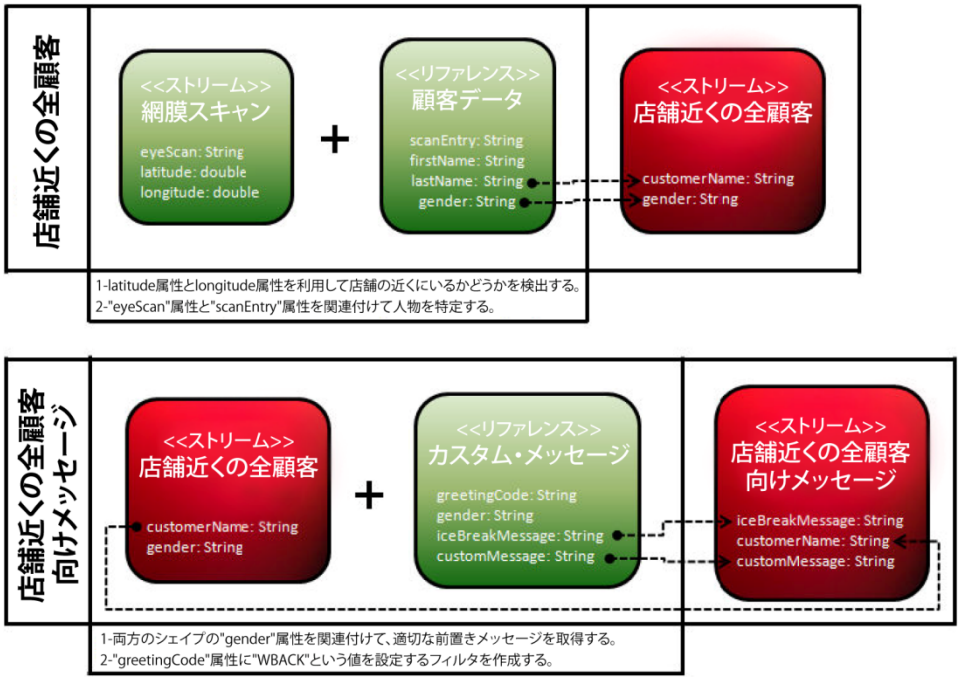

図5.ビジネス・ルールの説明を付けた最終版の概念モデル。

図 5 に細部を記述した概念モデルは、事例の実装に対して確実な基盤を提供します。ほとんとの場合、概念モデルはビジネス・シナリオの実装中に頭の中で作り込まれます。ここで示す手順は、ソリューション設計手法としての使用を意図したものではありません。代わりに、一般的なイベント処理アプリケーションの設計方法とこのようなソリューションでよく持ち上がる懸案事項を紹介しています。

第2部：Oracle Stream Explorerでのアーティファクトの実装

Oracle Stream Explorer でのアーティファクトの実装を始める前に、前提条件が満たされていることを確認する必要があります。第一に、DDL および DML 権限の付与された実行中データベースを使用できる必要があります。標準でサポートされているデータベースは Oracle、SQL Server 2005、Derby です。JDBC に準拠したその他のデータベースも使用できますが、JDBC ドライバのデプロイが必要になる場合があります。次に、SQL スクリプトを実行して、実装中に使用する 2 つのデータベース表を作成してデータを挿入する必要があります。この SQL スクリプトは本書の付録 A に記載されています。最後に、網膜スキャン・ストリームのアーティファクトを実装する際、CSV ファイルを使用します。このファイルの中身のサンプル・データと完全なファイルをダウンロードするための URL も付録 A に記載されています。

Oracle Stream Explorer でデータベース表のリファレンスとして使用する 2 つのソースを作成する必要があることから、データベース接続プールをセットアップしておく必要があります。データベース接続プールをセットアップできるようにするには、Oracle Event Processing Visualizer の管理者アクセスが必要です。Oracle Stream Explorer の実行中に、Web ブラウザを開いて URL（http://host:port/wlevs）にアクセスします。この URL で、host はサーバーが稼働する IP アドレスまたはホスト名であり、port は外部アクセス用に構成された HTTP 対応ポートになります。図 6 に Oracle Event Processing Visualizer のログイン画面を示します。

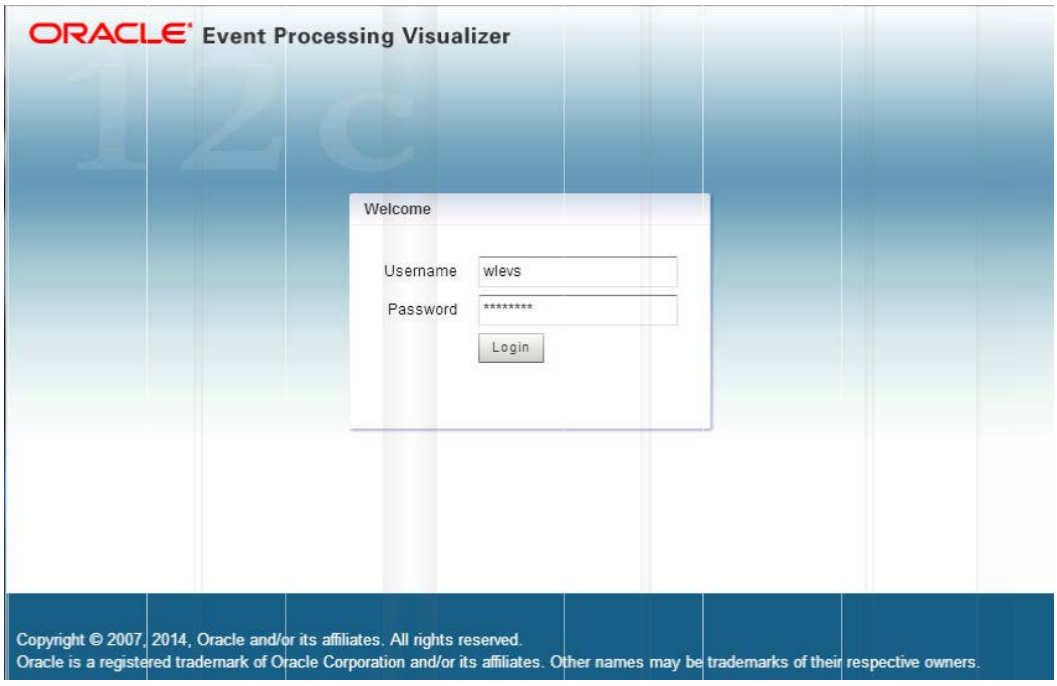

図6.Oracle Event Processing Visualizerのログイン画面

このコンソールを使用するには適切な資格証明が必要です。資格証明がない場合は、Oracle Stream Explorer 管理者に作成を依頼します。コンソールにログインしたら、Oracle Stream Explorer が稼働

するサーバー・インスタンスにアクセスします。ウィンドウが開き、すべてのサーバー情報がタブに表示されます。データベース接続プールを作成するには、はじめに「DataSource」タブをクリックします（図 7 を参照）。

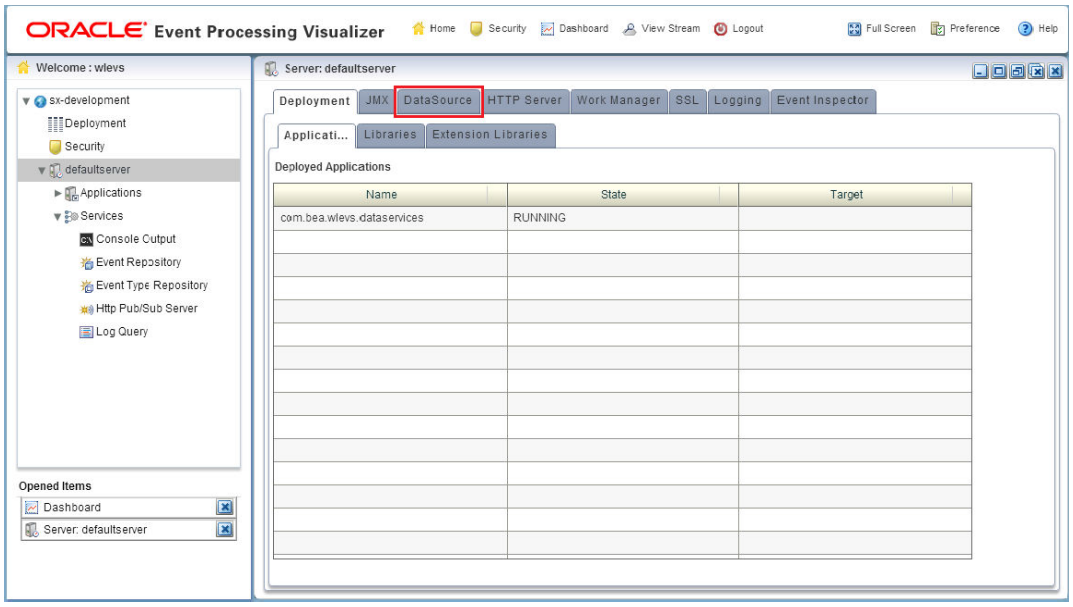

図7.コンソールでのデータベース接続プールの作成の開始

DataSource タブには、このサーバーに対してすでに作成済みのデータベース接続がすべて表示されます。「Add」ボタンをクリックして新規データベース接続プールを作成します。新規データソース作成ウィザードが開始されます。図 8 のように Name フィールドと JNDI Name フィールドの両方に"jdbc/minorityReport"と入力します。

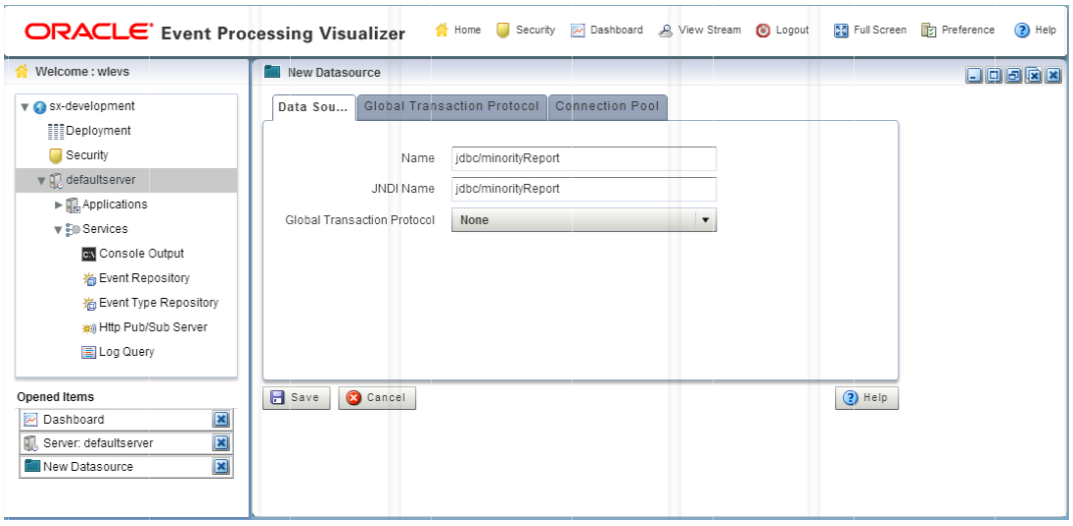

図8.データベース接続プールの作成の開始

図 8 に示したとおり、Global Transaction Protocol フィールドで「None」を選びます。これは非常に重要です。ここでの目的はデータベース表のデータを読み取るだけであるため、このフィールドに"None"を設定することでデータベースのオーバーヘッドを軽減できます。次に、「Global Transaction Protocol」タブをクリックして、データベースの JDBC 接続情報を入力します。

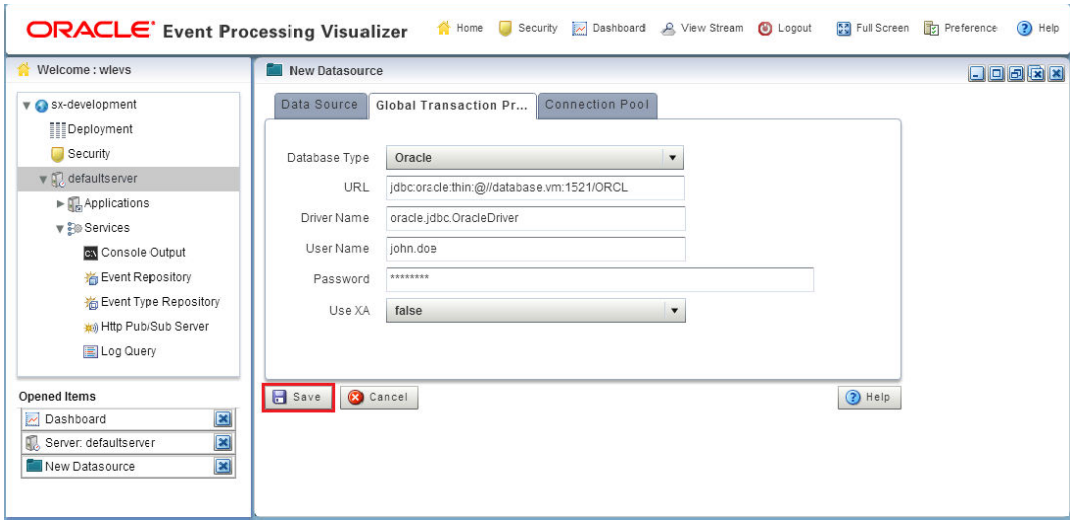

図9.データベースのJDBC接続情報の入力

この実装に必要な 2 つの表を含むデータベースについて必要な JDBC 接続情報を入力します。使用するデータベースの種類にかかわらず、Use XA フィールドの値に false を設定します。データベースの JDBC 接続情報の設定が終わったら、「Save」ボタンをクリックしてデータベース接続プールの作成を終了します。図 9 に構成の例を示しています。

データベース接続プールが正しく作成されたら、アーティファクトの開発を開始します。URL（http://host:port/sx）から Oracle Stream Explorer アプリケーションにアクセスします。ホストとポートは Oracle Event Processing Visualizer へのアクセスで使用したものと同じであり、資格証明も同じものを使用できます。図 10 に Oracle Stream Explorer のログイン画面を示します。

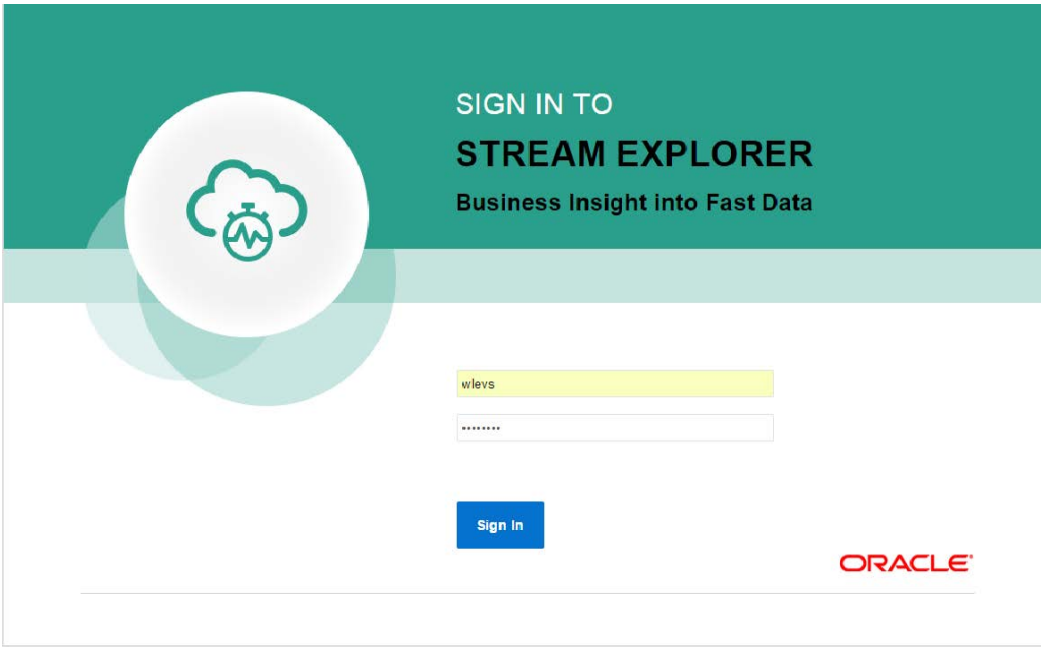

図10.Oracle Stream Explorerのログイン画面

Oracle Stream Explorer に対する認証が完了すると、ホーム画面が表示されます。ホーム画面にはウェルカム・メッセージが示され、イベント処理アプリケーションが使用される可能性がある一般的なビジネス・ドメインを表す枠内に、それぞれ趣向を凝らしたイメージが表示されます。いずれかのイメージをクリックするとビジネス・ドメインのタグが設定され、カタログ画面に直接移動します。ホーム画面の右上に表示される Catalog ボタンを使用すると、ビジネス・ドメイン・タグを設定せずにカタログ画面にアクセスできます。

Oracle Stream Explorer のアーティファクトを作成および保守するのは、このカタログ画面です。これまでに作成されたすべてのアーティファクトが画面中央のメイン表内部に表示され、エクスプロレーションやストリーム、リファレンスなどの特定のアーティファクトに対するフィルタリングを実行できます。アーティファクトを作成するには、Create New Item コンボ・ボックスを使用して新しいアーティファクトの作成をリクエストします（図 11 を参照）。

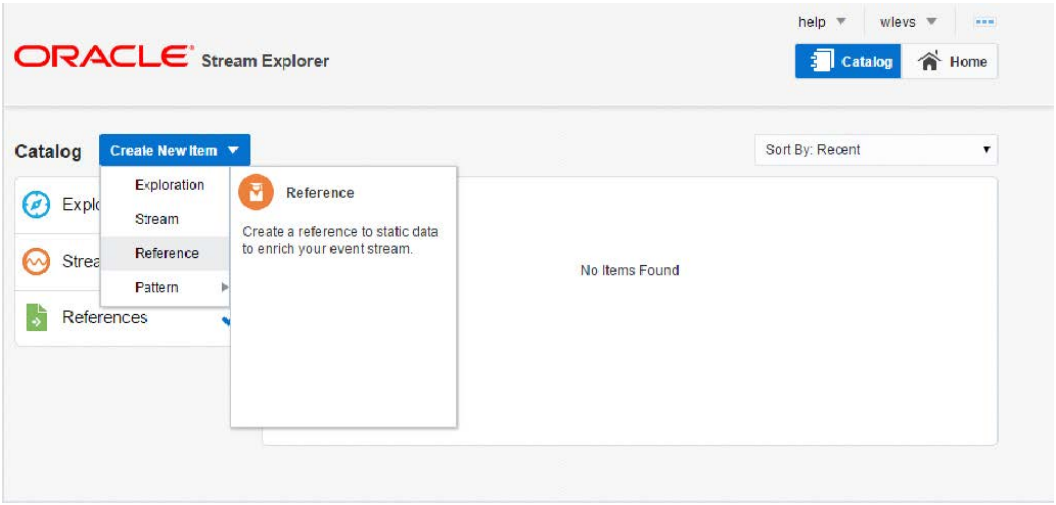

図11.Oracle Stream Explorerでの新規アーティファクトの作成リクエスト

最初に作成するアーティファクトは顧客リファレンスです。図 11 に示したとおり、リストから「Reference」を選んで新規リファレンスの作成をリクエストします。新規リファレンス作成ウィザードが開始されます。Name フィールドに"CustomerData"と入力します。Tags フィールドに"customer"と入力し、Source Type コンボ・ボックスで「Database Table」を選択します。図 12 にウィザードの最初のステップを示します。

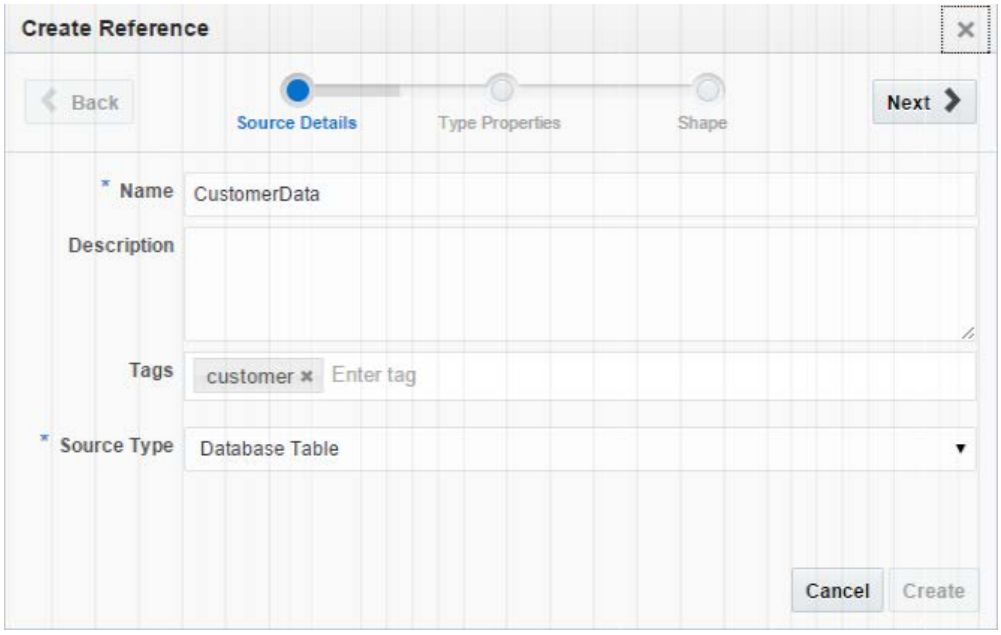

図12.新規リファレンス作成ウィザードの最初のステップ

「Next」ボタンをクリックして、2 番目のウィザード・ステップに進みます。データベース・サーバーへの接続に使用するデータベース接続プールの名前を入力します。図 13 に示すとおり、Data Source Name コンボ・ボックスで「jdbc/minorityReport」を選択します。「Next」ボタンをクリックして、3 番目のステップに進みます。

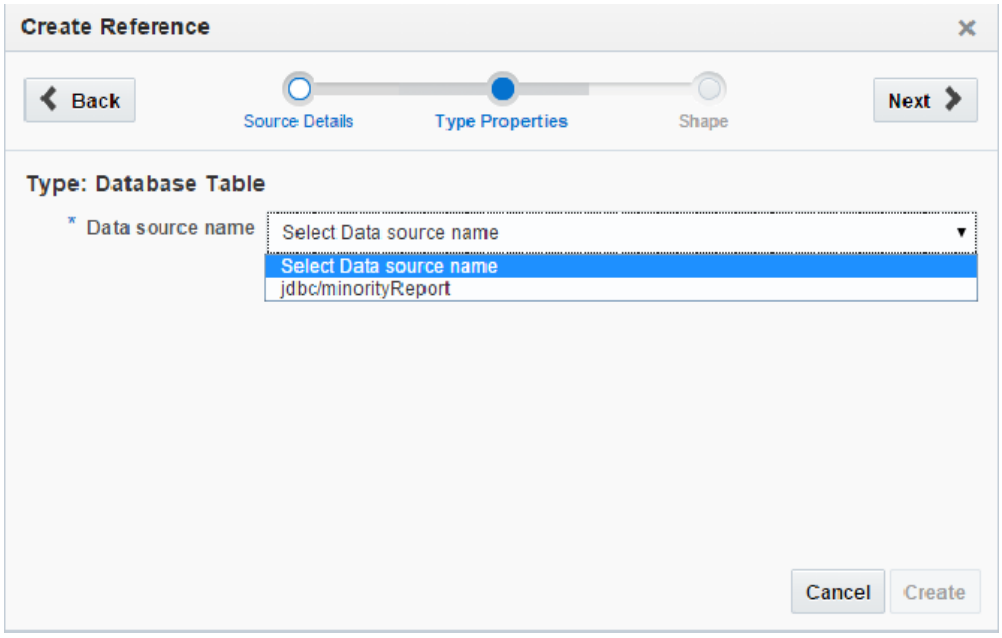

図13.新規リファレンス作成ウィザードの2番目のステップ

最後になる 3 番目のステップで、リファレンスが参照するデータベース表の名前を入力します。図 14 に示すとおり、Select Shape コンボ・ボックスで「CUSTOMER_DATA」を選択します。「Create」ボタンをクリックして、新規リファレンス作成ウィザードを終了します。

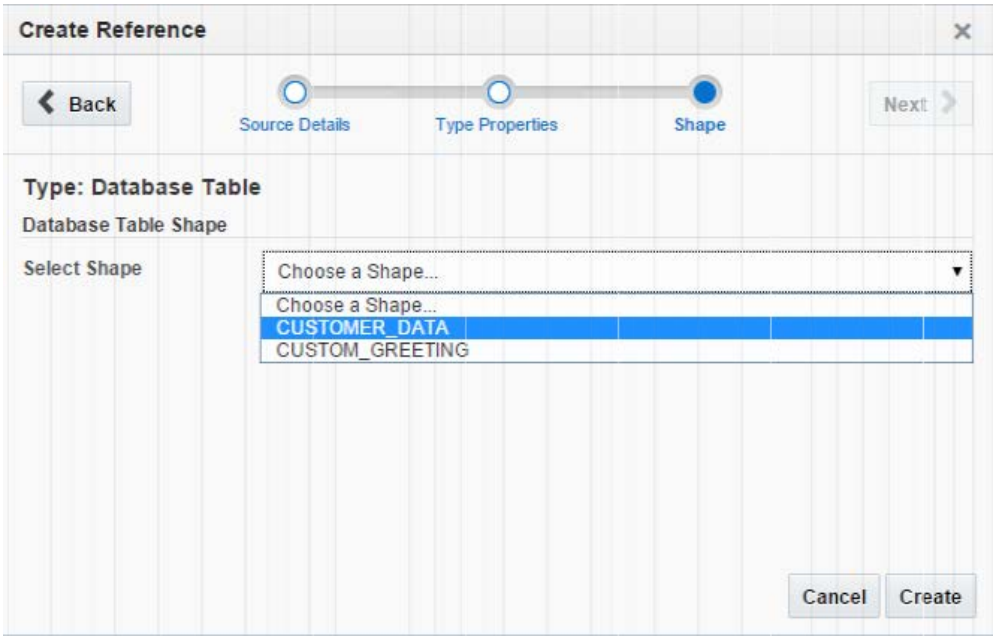

図14.新規リファレンス作成ウィザードの3番目のステップ

ウィザードが終了すると、"CustomerData"という名前の新しいシェイプが画面中央の表に表示されます。次に、同じ手順を使用して 2 番目のリファレンスを作成します。2 番目のリファレンスを作成するとき、Name フィールドに"CustomGreeting"を入力し、Select Shape では「CUSTOM_GREETING」データベース表を選択します。図 15 に、リファレンスを 2 つ作成した後のカタログ画面を示します。

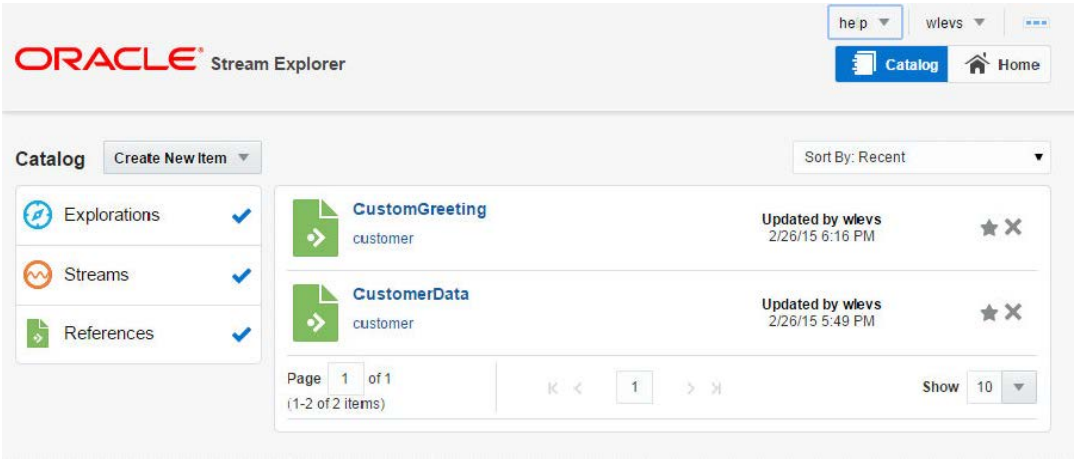

図15.作成された2つのリファレンスを表示したOracle Stream Explorer

ここで 3 番目のソースを作成しますが、今回はリファレンスではなくストリームを作成します。このストリームは網膜スキャンの受信フローを表し、対象人物の網膜データとその位置を含みます。図 11 に示したとおり、リストから「Stream」を選んで新規ストリームの作成をリクエストします。新規ストリーム作成ウィザードが開始されます。

Name フィールドに"EyeScanStream"と入力します。Tags フィールドに"customer"と入力し、Source Type コンボ・ボックスで「CSV File」を選択します。すべてのエクスプロレーションを順番に作成するため、「Create Exploration with this Source」チェック・ボックスはオフのままにします。「Next」ボタンをクリックして、2 番目のウィザード・ステップに進みます。ソース・タイプとして CSV を選んだため、CSV ファイルのソース・パスを入力する必要があります。「Upload File」ボタンをクリックして CSV ファイルを選択し、Oracle Stream Explorer にアップロードします。

「Next」ボタンをクリックして、3 番目のステップに進みます。Name フィールドにもう一度"EyeScanStream"と入力します。ファイルのアップロード・プロセス中にウィザードは CSV ファイルを解析し、属性名とデータ型を検出して、3 番目のウィザード・ステップで表示できます。このため、「Manual Mapping」フィールドが選択されており、表示された属性が図 16 に示すものと一致することを確認します。「Create」ボタンをクリックして、新規ストリーム作成ウィザードを終了します。

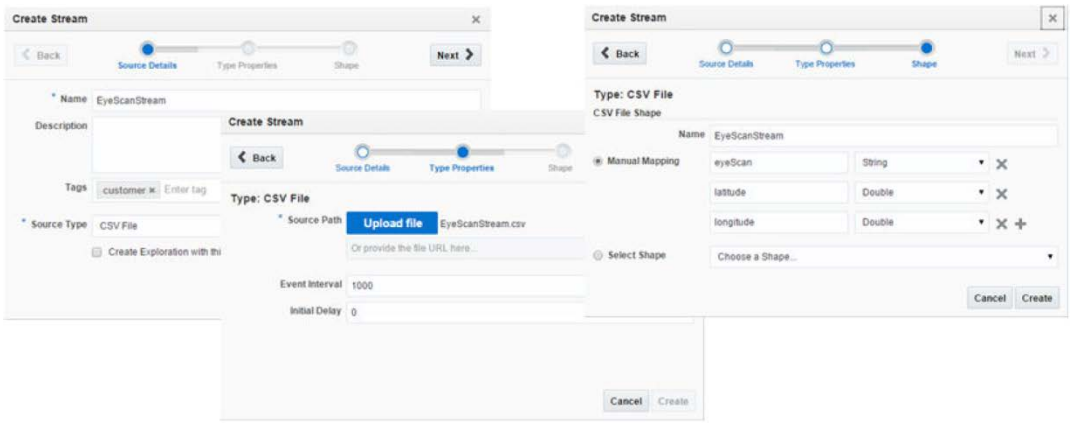

図16.新規ストリーム作成ウィザードの3つのステップ

ここまでに、事例の実装開始に必要なすべてのソースを作成しました。次に、エクスプロレーションの作成を開始します。エクスプロレーションは期待される動作を提供するアーティファクトです。最初に構築するエクスプロレーションは店舗に近づいた人物を検出し、顧客リファレンスと照合します。このエクスプロレーションには"All Customers Near the Store"（店舗近くの全顧客）という名前を付けます。図 11 に示したとおり、リストから「Exploration」オプションを選んで新規エクスプロレーションの作成をリクエストします。新規エクスプロレーション作成ウィザードが開始されます。Name フィールドに"All Customers Near the Store"と入力します。図 17 に示すとおり、Tags フィールドに"customer"と入力し、Source コンボ・ボックスで「EyeScanStream」を選択します。「Create」ボタンをクリックして、新規エクスプロレーション作成ウィザードを終了します。図 18 に示すとおり、エクスプロレーション・エディタに新しく作成したエクスプロレーションが表示されます。

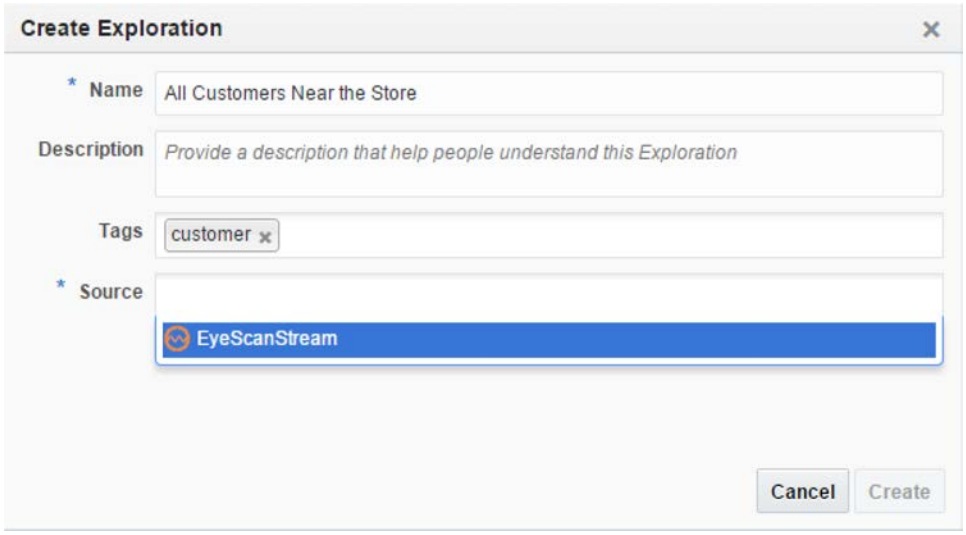

図17.新規エクスプロレーション作成ウィザードでのソースの設定

エクスプロレーション・エディタで最初に目に留まるのは、Live Output Stream セクションと Charts セクションであり、ソースから取得した結果（ここでは網膜スキャン・ストリーム）がリアルタイムで表示されます。ここで興味深いのは、エクスプロレーションが変化すると、これらのセクションもリアルタイムで更新されることです。そのため、出力結果がどうなるかについてユーザーは把握しやすくなります。図 18 に示す新しく作成したエクスプロレーションでは、ソースから取得したデータがリアルタイムで表示されます。

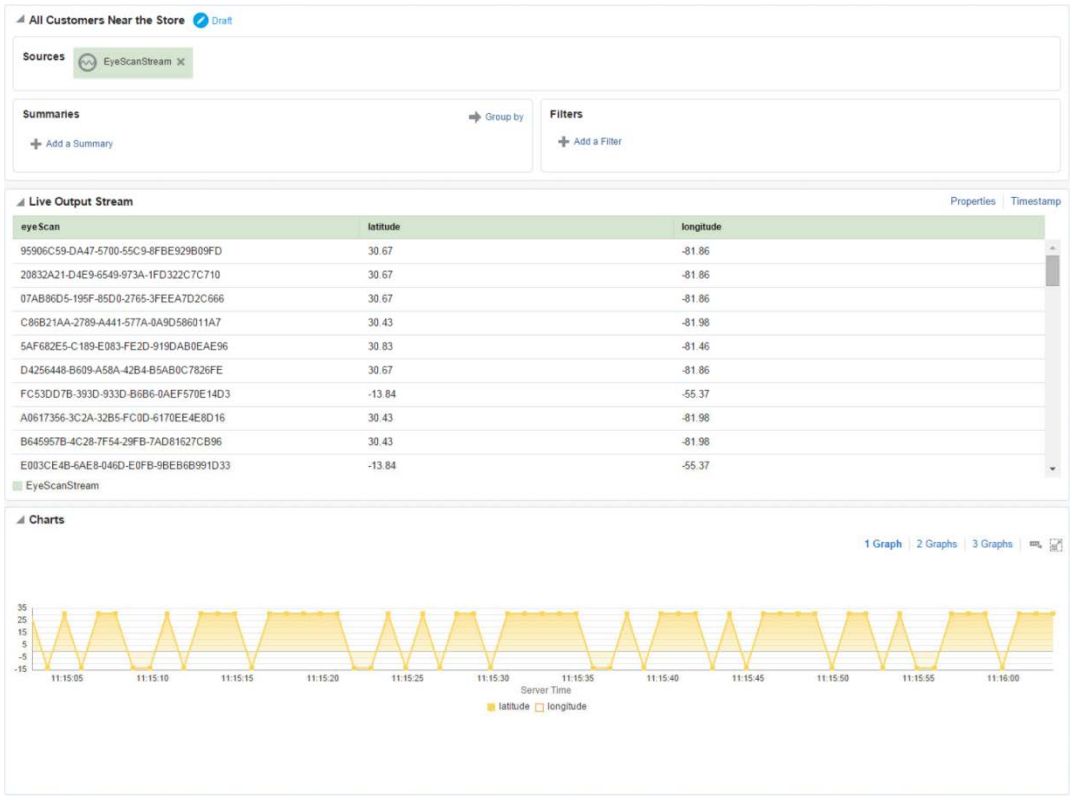

図18.新しく作成したエクスプロレーションによる、ソースから取得したデータのリアルタイム表示

このエクスプロレーションは店舗近くにいる人物のみを検出する必要があるため、対象となる人物のみを選び出すフィルタを作成する必要があります。Filters セクションの「Add a Filter」リンクをクリックします。新しく作成されたフィルタ・エントリで、処理対象の属性と使用する演算子を選び、求める出力結果を制限する値を設定します。このエクスプロレーションでは、フィルタによって位置の latitude 属性が"20.00"以上で、longitude 属性が"60.00"以下である人物のみが返されます。図 19 に、Oracle Stream Explorer で作成中の 2 つのフィルタを示します。

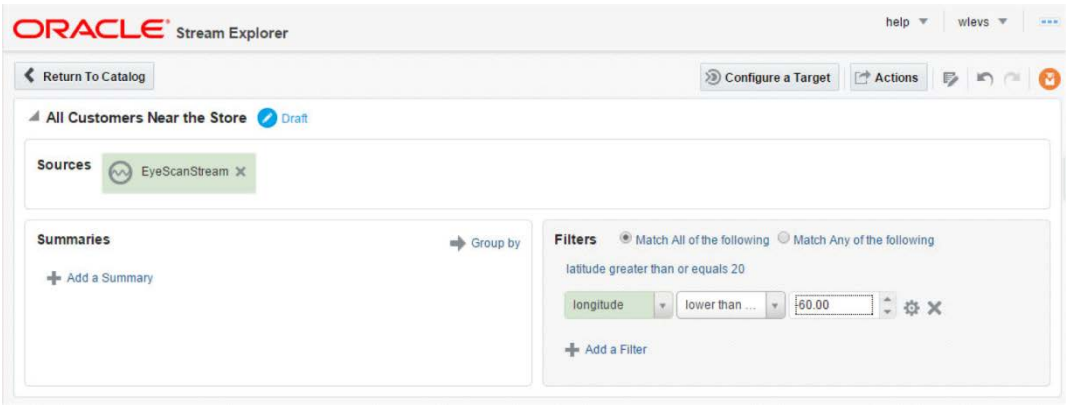

図19.Oracle Stream Explorerでの位置のフィルタリング

これらのフィルタを設定すると、Live Output Stream セクションで基準を満たさないイベントが自動的に削除され、Charts セクションのグラフがより平坦になります。これは、latitude 属性と longitude 属性の近似値のみがライブ出力ストリームに表示されるためです。

人物を特定するには、網膜スキャン・ストリームから取得したイベントに顧客情報を含む顧客リファレンスを関連付ける必要があります。はじめに、エクスプロレーションの Sources セクションを変更して、シェイプをもう 1 つ含めます。「Sources」セクションをクリックし、ドロップダウン・リストから「CustomerData」を選択します。1 つのエクスプロレーション内で複数のシェイプが使用されると、Correlations という新しいセクションがエクスプロレーション・エディタ内に表示されます。

「Add a Correlation」リンクをクリックして、ソース間に新しい相関を作成します。左側で「eyeScan」属性を選び、右側で「scan_entry」属性を選びます。相関の構成が完了したら、図 20 に示すように Live Output Stream セクションに 2 つのソースから取得した属性が表示されます。

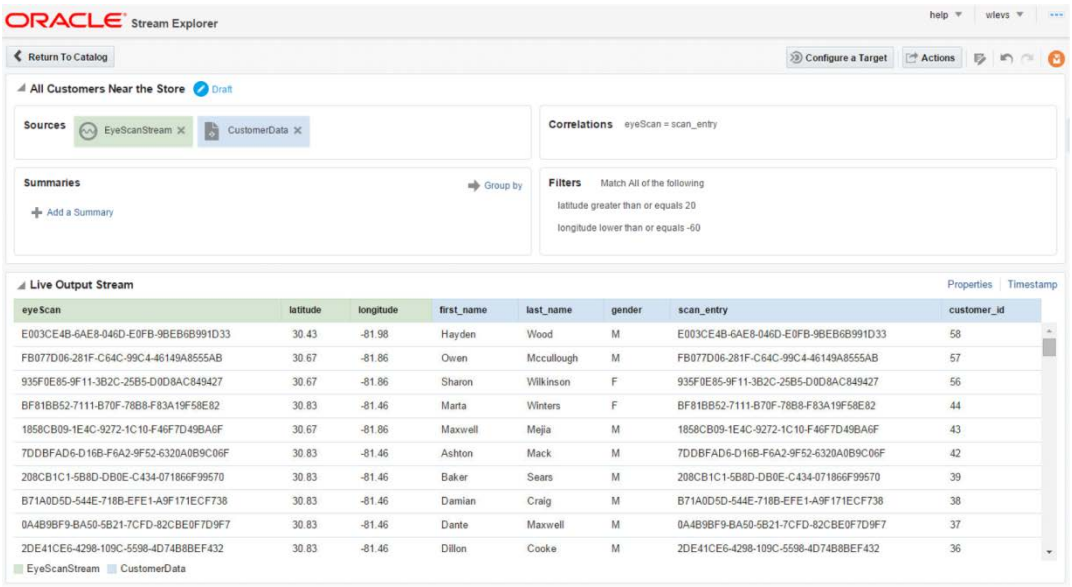

図20.網膜スキャン・ストリームと顧客リファレンス間のイベント相関の結果

エクスプロレーションが完了したと見なされるためには、期待される出力結果を提供する必要があります。これは、いったん公開されたエクスプロレーションは別のエクスプロレーションでソースのリストで使用される可能性があるためです。Oracle Stream Explorer では、Properties リンクから出力結果を変更できます。「Properties」リンクをクリックし、"last_name"属性と"gender"属性のみを選択した状態にして、これらの属性が properties セクションで 1 番上と 2 番目になるように順序を変更します。次に、カスタム表示名を入力するために「last_name」属性をダブルクリックし、値を"customerName"に設定します（図 21 を参照）。

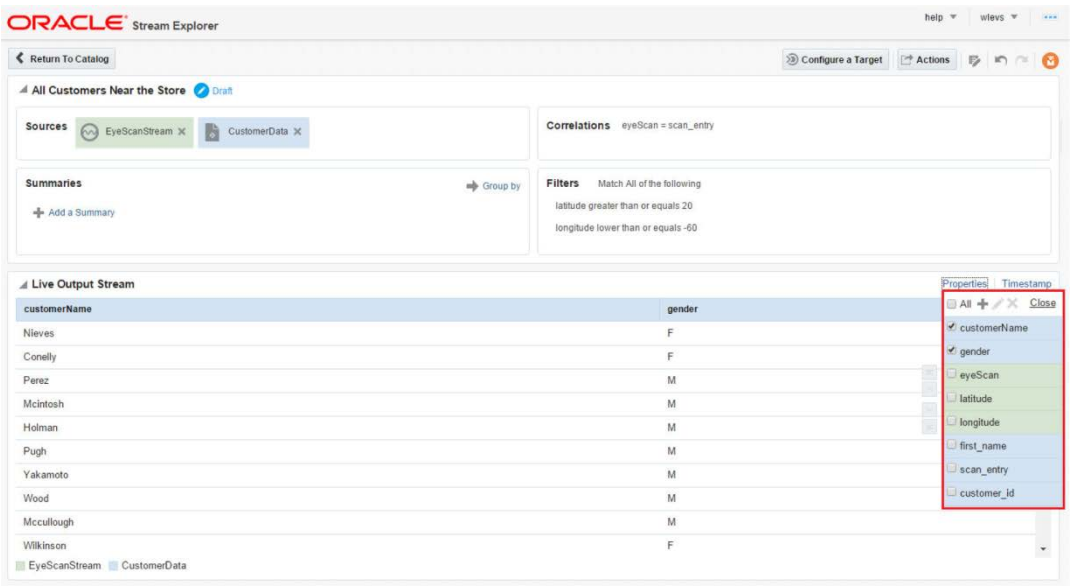

図21.公開前のエクスプロレーションに対する出力結果のカスタマイズ

以上で、エクスプロレーションを公開する準備が整いました。エクスプロレーション・エディタの右上にある「Actions」ボタンをクリックすると、エクスプロレーションで実行できるアクションに関連するボタンを含むメニューが表示されます。図 22 に示すように「Publish」ボタンをクリックします。

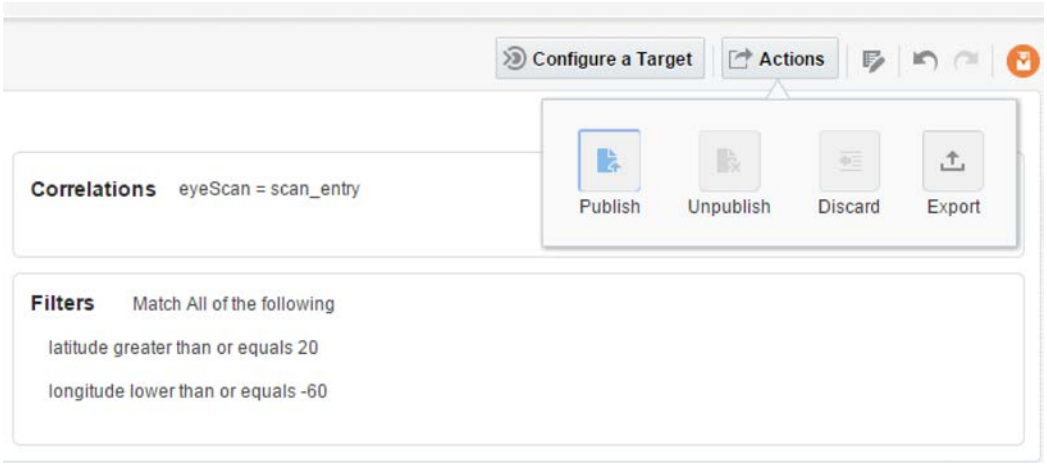

図22.ストリームとして使用できるようにするためのエクスプロレーションの公開

最初のエクスプロレーションを正しく公開したら、2 番目のエクスプロレーションの構築を始めます。2 番目のエクスプロレーションは店舗近くにいる顧客に対してメッセージを作成し、人物の性別に基づいて、使用すべき適切なメッセージを決定します。このエクスプロレーションには "Greetings for All Customers Near"（店舗近くの全顧客向けメッセージ）という名前を付けます。図 11 に示したとおり、リストから「Exploration」オプションを選んで新規エクスプロレーションの作成をリクエストします。新規エクスプロレーション作成ウィザードが開始されます。

Name フィールドに"Greetings for All Customers Near"と入力します。Tags フィールドに"customer"と入力し、Source コンボ・ボックスで「All Customers Near the Store」を選択します。「Create」ボタンをクリックして、新規エクスプロレーション作成ウィザードを終了します。エクスプロレーション・エディタに新しく作成したエクスプロレーションが開き、Live Output Stream セクションに 1 番目のエクスプロレーションによる出力結果が表示されます。「Sources」セクションをクリックし、ドロップダウン・リストから「CustomGreeting」を選択します。エクスプロレーション・エディタに Correlations セクションが表示されたら、「Add a Correlation」リンクをクリックしてソース間に新しい相関を作成します。左側で「gender」属性を選び、右側でも「gender」属性を選びます。相関の構成が完了したら、図 23 に示すように Live Output Stream セクションに 2 つのソースから取得した属性が表示されます。

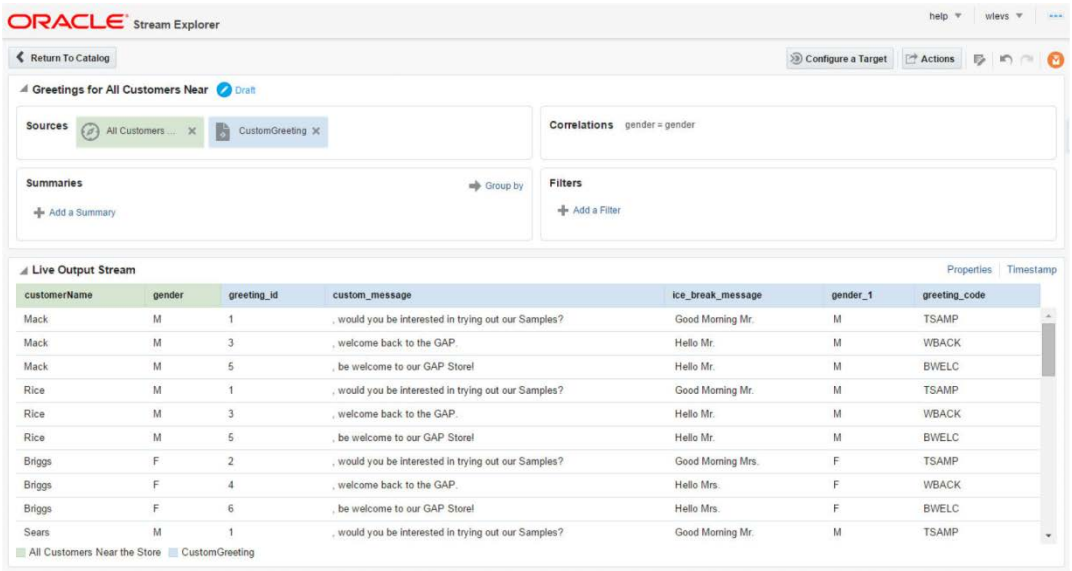

図23.相関が適用された2番目のエクスプロレーション

不要なメッセージを顧客に表示しないようにするため、フィルタを作成して"Welcome back to the GAP"メッセージのみが使用されるようにします。Filters セクションで「Add a Filter」リンクをクリックして新しいフィルタ・エントリを作成し、"greeting_code"属性の値が"WBACK"に等しくなるように設定します。

最初のエクスプロレーションと同様に、2 番目のエクスプロレーションでも出力結果をカスタマイズする必要があります。「Properties」リンクをクリックし、"ice_break_message"、"customerName"、"custom_message"属性のみを選択した状態にして、これらの属性が properties セクションで 1 番上から 3 番目になるように順序を変更します。引き続き Properties リンクで、3 つの属性にそれぞれカスタム表示名を設定します。図 24 に示すとおり、最初の属性には"iceBreakMessage"を、2 番目の属性には"customerName"を、3 番目の属性には"customMessage"を指定します。

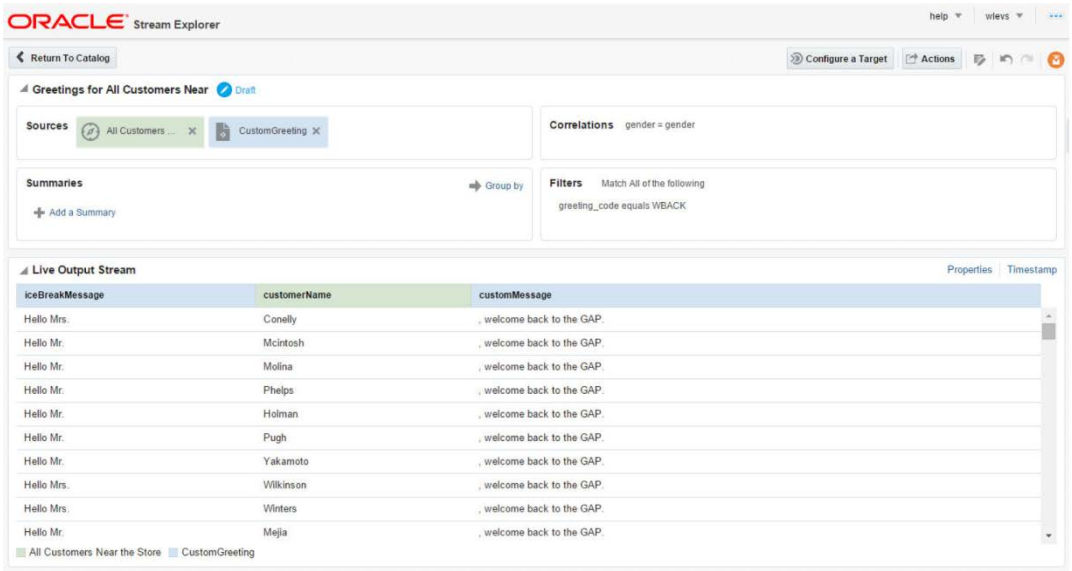

図24.出力結果を適切にカスタマイズした2番目のエクスプロレーション

図 22 に示した方法で 2 番目のエクスプロレーションを公開します。ここまでで事例の実装は完了しました。店舗に近づいた人がいると、Oracle Stream Explorer がリアルタイムでこれを検出し、カスタム・メッセージで出迎えます。また、自動化アプローチは人物の性別を使って男性か女性か見分けることで、顧客の名前の前に"Mr."や"Mrs."などの英語の敬称を付けます。

Oracle Stream Explorer のエクスプロレーション・エディタに正しい出力結果を表示することは興味深いことですが、挨拶メッセージを伝える外部システムにこの出力結果を送信できないとしたら、使いものにならないと言えるでしょう。Oracle Stream Explorer では Configure a Target ボタンを使用して、エクスプロレーションの出力結果を外部システムに送信できます。Oracle Stream Explorer で現在サポートされているターゲットは次のとおりです。

» **CSV File**：出力結果を CSV ファイルに書き出します。コンテンツを追加できます。
» **HTTP Publisher**：OEP からアウトバウンド HTTP チャネルに対して出力結果を公開します。
» **Event-Driven Network**：SOA Suite の EDN 対応サブスクライバに対して出力結果を送信します。
» **Java Message Service**：`javax.jms.MapMessage` を作成して JMS 宛先に送信します。
» **REST Endpoints**：REST エンドポイントに対して出力結果を指定した HTTP POST を実行します。

どのターゲット・タイプを選んだ場合も、設定されたターゲットに対して生成された出力結果が連続的に送信されます。つまり、Oracle Stream Explorer で新しい出力結果が使用可能になり次第、遅延なしで即座にこれが送信されます。このアプローチでは、該当する状況になったその瞬間に行動を起こせるため、真にイベント駆動型のアプローチが実現します。

たとえば、Java で実装された音声対応システムが JMS 経由で挨拶リクエストをリスニングし、MDB（Message-Driven Bean）を使用して JMS 宛先からのメッセージを消費し、Java Speech API を介して挨拶スピーチを実行するとしましょう。図 25 に示すようにターゲット・タイプに JMS を構成することで、エクスプロレーションによる出力結果を簡単に JMS 経由で送信できます。

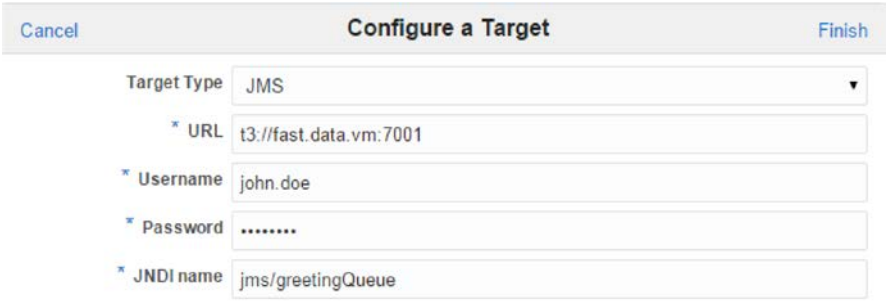

図25.出力結果を送信するためのJMSベース・ターゲットの構成

本書は Oracle Stream Explorer の基礎とイベント処理アプリケーションの構築方法に焦点を合わせているため、Java を使用した音声対応システムの完全な実装については、言うまでもなく対象外になります。ただし、Java Speech API を介して人声を使用した挨拶を合成する MDB の実装方法を付録 B に記載しています。

結論

バックミラーだけを見ながら自動車を運転することを想像できますか。数多くの企業はこの方法で事業を運営し、従来のデータウェアハウスとビジネス・インテリジェンス・テクノロジーを使用して知見を得ようとしています。過去の情報を見るだけでは必ずしも適切な対応を取ることはできません。この場合、重要な機会を見過ごしたり、脅威となる存在が認識不足につけ込んだりする可能性があります。

Web ベースのアプリケーションである Oracle Stream Explorer は、Oracle Event Processing の機能を活用して、イベント・ストリームをリアルタイムに分析するためのツールをビジネス・ユーザーに提供します。これにより、ビジネス・ユーザーは知見を獲得し、必要に応じて対策を講じることができます。本書では Oracle Stream Explorer の基礎を紹介するとともに、イベント処理ベースのアプリケーション開発方法を事例の詳しい実装ステップを通じて説明しました。

付録A：事例で使用したスクリプトとサンプル

本書に記載した事例を実装できるようにするには、2 つのデータベース表を作成する必要があります。リスト 1 に示すスクリプトは 2 つのデータベース表を作成してサンプル・データを移入します。このスクリプトは Oracle データベースに対して正しく動作することがテストされていますが、その他のデータベースでも動作する場合があります。表の構造と列名を変更しない限り、必要に応じてスクリプトを調整することができます。

```
--------------------------------------------------------
-- DDL for the table CUSTOMER_DATA
--------------------------------------------------------

CREATE TABLE "CUSTOMER_DATA"
(    "CUSTOMER_ID" INTEGER NOT NULL PRIMARY KEY,
     "SCAN_ENTRY" VARCHAR2(255) NOT NULL,
     "FIRST_NAME" VARCHAR2(20) NOT NULL,
     "LAST_NAME" VARCHAR2(20) NOT NULL,
     "GENDER" VARCHAR2(1) NOT NULL
)
```

```
-------------------------------------------------------
--  DDL for the table CUSTOM_GREETING
-------------------------------------------------------

  CREATE TABLE "CUSTOM_GREETING"
   (    "GREETING_ID" INTEGER NOT NULL PRIMARY KEY, "GREETING_CODE" VARCHAR2(5) NOT
        NULL, "GENDER" VARCHAR2(1) NOT NULL, "ICE_BREAK_MESSAGE" VARCHAR2(50) NOT
        NULL, "CUSTOM_MESSAGE" VARCHAR2(100) NOT NULL
   );

-------------------------------------------------------
--  Loading data into the table CUSTOMER_DATA
-------------------------------------------------------

Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (1,'CB281A82-EBF9-247F-8D4C-632F2CD4DC9E','Gregory','Berger','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (2,'B7C841D6-DC7D-0C32-1402-58E0E06F0C74','George','Griffith','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (3,'0028CF68-2264-D591-C3F8-EECF78E3635F','Damian','Key','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (4,'B5C9DA94-6AF7-9BFF-19F5-9EECC4797417','Scott','Bridges','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (5,'1EB943F4-C7B3-29D8-D0D3-042507CCDD50','Colton','Kinney','F');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (6,'B71E107F-DD89-76EC-497B-9F3BB81CC790','Brody','Goodman','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (7,'AF25D32D-0870-7C79-5ECE-A8D88E63A099','Simon','Park','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (8,'EA712817-9B59-042A-F952-4BD9B2621FE7','Addison','Medina','F');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (9,'27AFC1DC-74D6-D988-9E7E-35522FD65CAC','Charles','Sutton','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (10,'45D8A1B8-72DB-A6EE-7725-4AA8F379FA2E','Vaughan','Lawson','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (11,'3FAB4B01-AA8C-CA14-03B7-69F75670D0C2','Felix','Haley','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (12,'F6C6D0BC-5772-97BA-2EB3-E0A6DE91D64B','Mason','Mclean','F');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (13,'096458CC-3F58-0CFC-08B9-08E2134AFCB7','Brent','Faulkner','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (14,'95005563-5DFB-2F6D-5B12-1BB0DEE71492','Kasimir','Fitzgerald','F');
```

```
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (15,'F812B9FD-0EC3-30FE-3D3D-37ECB7A5B012','Zahir','Savage','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (16,'5F8F72F2-48DF-2507-6D34-AA2E6FD663B8','Jin','Wilcox','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (17,'C54C5779-7BB8-4B02-D78E-153879C9876A','Chadwick','Snyder','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (18,'D1EB489E-3FA6-C4CC-F923-CFB6005447F6','Chaim','Moses','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (19,'50035120-B48A-4904-5498-591F6F15A1FD','Lester','Fitzgerald','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (20,'F35FB6B7-6B68-4E9E-5D1F-7EE852A11F8A','Otto','Dixon','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (21,'DBA752B4-5ABE-BEB5-360C-572E52F02B4B','Reese','Gross','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (22,'23E5896B-9F6C-1E18-E671-3BABD8E41E05','Griffith','Fields','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (23,'480ED76A-D8DC-3C45-1304-EC7EECD18B25','Erasmus','Cobb','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (24,'FF7A36A3-C8B4-4305-9911-64F61FEC0CA7','Cooper','Barrett','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (25,'E083DDBE-282B-2CCF-7D8A-0C1D77D93E5A','Stephen','Rivas','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (26,'B7E4A6AD-6331-09F1-43D8-CA849A2FD796','Steven','Kramer','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (27,'3BB44589-FD5E-8CAB-5E76-7640897ACCBE','Kirsten','Malone','F');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (28,'D929DC13-0616-A3EC-0191-8CE9B725F37F','Perry','Bender','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (29,'B441636D-CDE4-A840-E848-B650635129C3','Alfonso','Velez','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (30,'B8D4A245-67F0-9386-7AAE-11F0E30C582C','Brody','Craft','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (31,'BFD9B89D-08E3-6DB9-8F76-A9499A2B50D8','Trevor','Hinton','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (32,'FC1E3D87-3955-7F7C-F865-EDD637D0558C','Nasim','Allison','F');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (33,'60473650-8AB0-12F4-9C91-84B9E3B86DE7','Ali','Solomon','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (34,'F44E0BF6-2F46-29D4-D9BC-4D35009C0D3B','Kennan','Schneider','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (35,'77F7D210-C655-21DA-B8EC-869ED54F820D','Andrea','Santos','F');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (36,'2DE41CE6-4298-109C-5598-4D74B8BEF432','Dillon','Cooke','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (37,'0A4B9BF9-BA50-5B21-7CFD-82CBE0F7D9F7','Dante','Maxwell','M');
```

```
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (38,'B71A0D5DD-544E-718B-EEFE1-A9F171ECFCF738','Damianan','Craig','M'M');
Insert into CUSTOMER_DATAA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (39,'208CB1C1-5B8D-DB0E-C434-071866F99570','Baker','Sears','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (40,'8873F976-89FC-24A3-71B5-A1AB6FE9D30E','Kelly','Briggs','F');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (41,'3E4F0B65-7445-1CD8-8A31-B5BAEED5CFA7','Eric','Rice','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (42,'7DDBFAD6-D16B-F6A2-9F52-6320A0B9C06F','Ashton','Mack','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (43,'1858CB09-1E4C-9272-1C10-F46F7D49BA6F','Maxwell','Mejia','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (44,'BF81BB52-7111-B70F-78B8-F83A19F58E82','Marta','Winters','F');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (45,'CA2582D4-40A0-C718-0F34-12AD26AAEB3B','Harlan','Wilcox','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (46,'0D333238-D10C-8FF9-B2EB-BCC1B1876767','Drew','Lynch','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (47,'656D6A68-7C70-02BC-32A8-240E5B48332D','Felix','Cash','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (48,'7158556F-CF03-05EA-D43F-AF2B2DA28BEE','Clayton','Chambers','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (49,'EA620BA5-CD44-2487-87EA-9CD352172BCC','Keaton','Woodward','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (50,'DD5FF2F5-ADE0-18F6-1E86-5E98C039D9B5','Ricardo','Ferreira','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (51,'7DBF9ABB-4979-6BE6-2BBE-A782512D6AF5','Oliver','Hurst','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (52,'F83662D2-BC5A-1E00-4910-53AE7FFB85C4','Channing','Hubbard','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (53,'4BA48031-B489-21BE-3EAD-C2E245AEDC21','Peter','Frazier','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (54,'CBD8BCD1-0385-C7F0-6D94-DAF8674087FA','Kieran','Farley','F');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (55,'617E92EB-DDA0-3200-C13C-CBAC5523526C','Monica','Atkins','F');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (56,'935F0E85-9F11-3B2C-25B5-D0D8AC849427','Sharon','Wilkinson','F');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (57,'FB077D06-281F-C64C-99C4-46149A8555AB','Owen','Mccullough','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (58,'E003CE4B-6AE8-046D-E0FB-9BEB6B991D33','Hayden','Wood','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (59,'B645957B-4C28-7F54-29FB-7AD81627CB96','Mitsuko','Yakamoto','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (60,'A0617356-3C2A-32B5-FC0D-6170EE4E8D16','Garrison','Pugh','M');
```

```
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (61,'FC53DD7B-393D-933D-B6B6-0AEF570E1E14D3','Salvadador','Holman','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (62,'D4256448-B609-A58A-42B4-B5AB0C7826FE','Tyrone','Phelps','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (63,'5AF682E5-C189-E083-FE2D-919DAB0EAE96','Ryan','Molina','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (64,'C86B21AA-2789-A441-577A-0A9D586011A7','Paul','Mcintosh','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (65,'07AB86D5-195F-85D0-2765-3FEEA7D2C666','Ralph','Perez','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (66,'20832A21-D4E9-6549-973A-1FD322C7C710','Jennifer','Conelly','F');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (67,'EC4EFB21-AFC6-A079-4FAE-9B6BE9EA62AC','Raymond','Gould','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (68,'12A4FECC-3BA8-1D1C-2B35-BB478BDAD662','George','Morse','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (69,'BF2D39C3-6030-6938-EEEA-4863600E7519','Matthew','Cole','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (70,'13E4712C-463F-BAA7-8537-A657BEB01D28','Blake','Benson','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (71,'830373DE-038B-95C5-467C-2573DFE51586','Debbie','Howell','F');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (72,'0D562E79-16B8-B81D-6144-913CA61DA166','Nataly','Shaffer','F');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (73,'BE643FCC-5F3B-E220-3670-FA99F7A1E507','Chandler','Dillon','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (74,'FD4483E9-928B-91C9-BDB1-A1A62FE2CA24','Hamilton','Rodriquez','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (75,'18555BC9-0B99-2F86-8FBB-9ED4B5FBF1FC','Jeff','Mcdaniel','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (76,'C32D18A2-85AA-F3A3-5FD1-8520D7892D40','Alfonso','Salazar','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (77,'EAB26AAA-DD16-0B56-4A6B-7BA332BCFB52','Calvin','Underwood','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (78,'375FC126-B176-AC8D-D31D-A589AA9B40EC','Maria','Nieves','F');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (79,'45BB2865-CAA5-29E4-3E9B-BE4010A3F9E2','Jonas','Sawyer','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (80,'B4D9B9DE-5912-CE87-235C-6CB165AD0FA0','Devin','Harper','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (81,'B1EBF0B6-F7C1-3DC4-3214-35E7F5238C47','Aquila','Hatfield','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (82,'5B69FE1B-2B9E-3919-F988-10E4298CF4F4','Derek','Richard','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (83,'FCFC70E9-6334-8B6C-D538-7161AFA80739','Chaney','Pratt','M');
```

```
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (84,'471AF640-42B5-2D28-F0057-9F3CFE2072A2','Malik','Beard','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (85,'332BF106-9B6B-C30C-1426-349909637024','Nathaniel','Cox','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (86,'6345109E-26FD-198B-F11F-410C93E52651','Elaine','Vargas','F');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (87,'8B111703-04FD-D5B6-1F9B-A1C5A5077241','Nissim','Macias','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (88,'1C7FBB69-266F-1F8D-902A-4F987115CABC','Hu','Stein','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (89,'95906C59-DA47-5700-55C9-8FBE929B09FD','Pamela','Hatfield','F');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (90,'BFBE734E-C80D-35C5-B441-139B7296712E','Cyrus','Gay','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (91,'B577A64A-C1ED-18C4-01CF-8F72EC65FA8C','Fletcher','Wiley','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (92,'FAFCE8D2-1500-8712-38A6-710BDE770A4A','Fitzgerald','Hughes','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (93,'4ED62A82-FDCE-445C-F12F-335745870917','Diane','Reese','F');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (94,'C89BD428-8B10-933B-CC58-5B01DDA827EA','Ingrid','Harrell','F');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (95,'1B5C67BF-33F1-9F7B-D94A-11560C15617C','Kenyon','Boyer','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (96,'1829958B-0CC7-0B93-B804-8F023188DC2F','Emerson','Mcgowan','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (97,'ECA02944-0A5F-023B-A292-223016BA2B3C','Berk','Simon','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (98,'4E8DEBDD-2E3E-B59E-D785-234767DC8522','Chancellor','Sears','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (99,'0519C7FC-3292-0B00-BB97-E0464F7D64ED','Grant','Williamson','M');
Insert into CUSTOMER_DATA (CUSTOMER_ID,SCAN_ENTRY,FIRST_NAME,LAST_NAME,GENDER)
values (100,'47CCF80B-46A0-66C8-BA1A-2DA5FB862AEE','Judah','Salazar','M');

------------------------------------------------------------
--  Loading data into the table CUSTOMER_DATA
------------------------------------------------------------

Insert into CUSTOM_GREETING
(GREETING_ID,GREETING_CODE,GENDER,ICE_BREAK_MESSAGE,CUSTOM_MESSAGE)   values
(1,'TSAMP','M','Good Morning Mr. ',', would you be interested in trying out our
Samples?');
Insert into CUSTOM_GREETING
(GREETING_ID,GREETING_CODE,GENDER,ICE_BREAK_MESSAGE,CUSTOM_MESSAGE)   values
```

```
(2,'TSAMP','F','Good Morning Mrs. ',', would you be interested in trying out our
Samples?');
Insert into CUSTOM_GREETING
(GREETING_ID,GREETING_CODE,GENDER,ICE_BREAK_MESSAGE,CUSTOM_MESSAGE)   values
(3,'WBACK','M','Hello Mr. ',', welcome back to the GAP.');
Insert into CUSTOM_GREETING
(GREETING_ID,GREETING_CODE,GENDER,ICE_BREAK_MESSAGE,CUSTOM_MESSAGE)   values
(4,'WBACK','F','Hello Mrs. ',', welcome back to the GAP.');
Insert into CUSTOM_GREETING
(GREETING_ID,GREETING_CODE,GENDER,ICE_BREAK_MESSAGE,CUSTOM_MESSAGE)   values
(5,'BWELC','M','Hello Mr. ',', be welcome to our GAP Store!');
Insert into CUSTOM_GREETING
(GREETING_ID,GREETING_CODE,GENDER,ICE_BREAK_MESSAGE,CUSTOM_MESSAGE)   values
(6,'BWELC','F','Hello Mrs. ',', be welcome to our GAP Store!');
```

リスト 1. データベース表を作成して移入するための SQL スクリプト

網膜スキャン・ストリームを実装するには、サンプル・データが移入された CSV ファイルを使用する必要があります。リスト 2 に示すのはファイルの一部であり、最初の 10 行のみが含まれています。この CSV ファイルは、3 つのフィールド（人物の網膜データ、位置の緯度および経度）が指定された各行で構成されます。

```
eyeScan,latitude,longitude
C86B21AA-2789-A441-577A-0A9D586011A7,30.831086,-81.460571 CB281A82-EBF9-247F-
8D4C-632F2CD4DC9E,30.425764,-81.975556 0A4B9BF9-BA50-5B21-7CFD-
82CBE0F7D9F7,30.425764,-81.975556 E003CE4B-6AE8-046D-E0FB-9BEB6B991D33,-
13.843414,-55.371094 B645957B-4C28-7F54-29FB-7AD81627CB96,-13.843414,-55.371094
B7C841D6-DC7D-0C32-1402-58E0E06F0C74,30.674122,-81.862946 0028CF68-2264-D591-
C3F8-EECF78E3635F,30.674122,-81.862946
5AF682E5-C189-E083-FE2D-919DAB0EAE96,-13.843414,-55.371094 C86B21AA-2789-A441-
577A-0A9D586011A7,-13.843414,-55.371094 1B5C67BF-33F1-9F7B-D94A-
11560C15617C,30.674122,-81.862946
```

リスト2.網膜スキャン・ストリームで使用されるCSVファイルの例

Oracle Stream Explorer を使用したテストでは、リスト 2 に示した例ではなく完全版のファイルを使用することを強く推奨します。完全版の CSV ファイルは次の URL からダウンロードできます。http://www.ateam-oracle.com/wp-content/uploads/2015/02/EyeScanStream.csv.

付録B：挨拶用Message-Driven Beanの作成

Oracle Stream Explorer を使用するとエクスプロレーションの出力結果を外部システムに送信できるため、この機能を使用してコンテキストに基づく方法で検知と反応を行うソリューションを考案すると面白いでしょう。たとえば、本書で実装した事例では、店舗近くを歩く人物に向けてカスタマイズした挨拶メッセージを作ることができます。しかし、エクスプロレーションによる出力結果を実際の挨拶に変換するとしたらどうなるでしょうか。

Java Speech API を使用すると、テキスト・メッセージを読み上げる音声ベース・システムを作成できます。当初は Sun Microsystems によって開発され、Apple、AT&T、IBM などの企業との協力を通じて完成した Java Speech API 仕様の最初のバージョンは 1998 年 10 月 26 日にリリースされました。技術的な面から見ると、Java Speech API は JDK 実装に含まれていないため、独自の実装を提供するサードパーティのスピーチ・ベンダーからの入手が必要です。もっとも一般的な実装の 1 つは、完全に Java で書かれたオープンソース・スピーチ・シンセサイザの FreeTTS です。

FreeTTS の実装を使用して、JMS 経由で挨拶リクエストをリスニングする音声ベース・システムを構築する方法を示すために、リスト 3 に、Java Speech API を介して人声を使用した挨拶を合成する MDB の例を示します。

```
package com.oracle.fmw.ateam.fastdata;

import javax.annotation.PostConstruct; import  javax.annotation.PreDestroy;
import javax.ejb.ActivationConfigProperty; import javax.ejb.EJBException;
import javax.ejb.MessageDriven; import javax.jms.MapMessage; import
javax.jms.Message;
import javax.jms.MessageListener;

import com.sun.speech.freetts.Voice;
import com.sun.speech.freetts.VoiceManager;

                        @MessageDriven(activationConfig = {
                            @ActivationConfigProperty(
          propertyName = "destinationType", propertyValue = "javax.jms.Queue"),
                            @ActivationConfigProperty(
          propertyName = "connectionFactoryJndiName", propertyValue =
          "jms/connFact"),
                            @ActivationConfigProperty(
          propertyName = "destinationJndiName", propertyValue =
          "jms/greetingQueue")})

public class GreetingListener implements MessageListener {

        private Voice voice;

        @PostConstruct
        private void allocateVoice() {

                VoiceManager voiceManager = VoiceManager.getInstance(); voice =
                voiceManager.getVoice("kevin16"); voice.allocate();

        }

        @PreDestroy
        private void deallocateVoice() {

                if (voice != null) {
```

```
                voice.deallocate();

        }

}

@Override
public void onMessage(Message message) {

        if (message instanceof MapMessage) {

                MapMessage mapMessage = null; String iceBreakMessage = null;
                String customerName = null; String custommessage = null;
                String greeting = null;

                try {

                        mapMessage = (MapMessage) message;

                        iceBreakMessage =
                        mapMessage.getString("iceBreakMessage");
                        customerName = mapMessage.getString("customerName");
                        custommessage =
                        mapMessage.getString("custommessage");

                        greeting = iceBreakMessage + customerName +
                        custommessage; voice.speak(greeting);

                } catch (Exception ex) {

                        throw new EJBException(ex);

                }

        }

}

 }
```

リスト3.Java Speech APIを使用したMDBの実装

リスト 3 に示した `onMessage()` メソッドの実装では、受信したメッセージが `javax.jms.MapMessage` に変換されています。これは、JMS ベースのターゲットを使用する場合に Oracle Stream Explorer が出力結果を送信するメッセージ・タイプだからです。また、図 24 に示したように、メッセージから取得した属性とエクスプロレーションで生成された属性が一致しています。

コンパイルの面から言うと、クラスパスで利用できる必要のある FreeTTS 実装のライブラリは $FREETTS/lib/freetts.jar ライブラリのみです。ただし、デプロイの面ではその他の FreeTTS 実装ライブラリもデプロイする必要があります。MDB と一緒に（EAR を使用してパッケージ化する場合）、または Java EE アプリケーション・サーバーのクラスパスに直接インストールします。デプロイ対象のライブラリは次のとおりです。

» $FREETTS/lib/cmulex.jar
» $FREETTS/lib/cmu_us_kal.jar
» $FREETTS/lib/en_us.jar
» $FREETTS/lib/freetts.jar

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

海外からのお問い合わせ窓口
電話：+1.650.506.7000
ファクシミリ：+1.650.506.7200

ORACLE

CONNECT WITH US

blogs.oracle.com/oracle
facebook.com/oracle
twitter.com/oracle
oracle.com

Hardware and Software, Engineered to Work Together

Oracle Stream Explorer の概要
2015 年 3 月
著者：Ricardo Ferreira
レビュー担当者：Peter Farkas、Prabhu Thukkaram

Oracle is committed to developing practices and products that help protect the environment