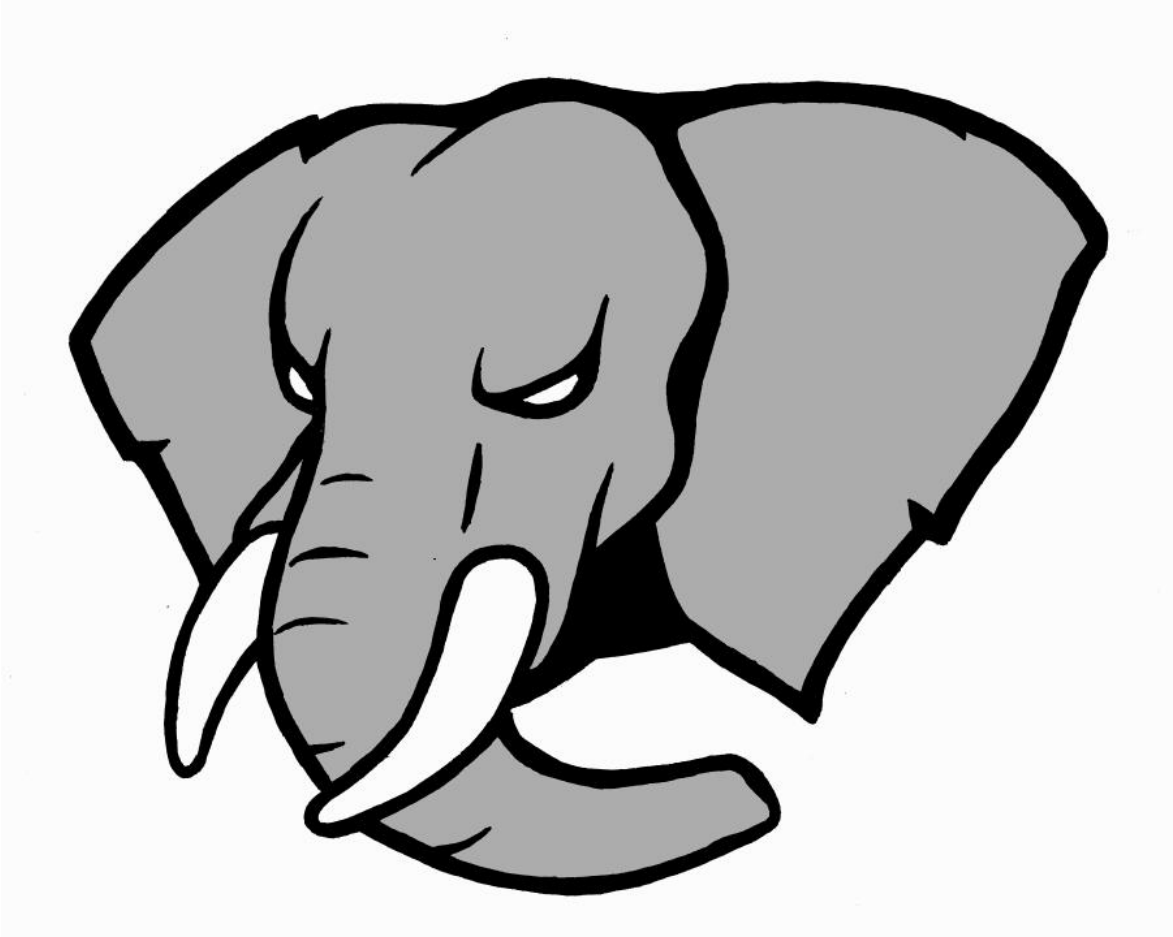

Slony-I
PostgreSQL のためのレプリケーションシステム

実装の詳細

Jan Wieck

Afilias USA INC.
Harsham, Pennsylvania, USA

概要

この文書は Slony-I レプリケーションエンジンと関連するコンポーネントのいくつかの実装の詳細を記載しています。

目次

1. データの操作

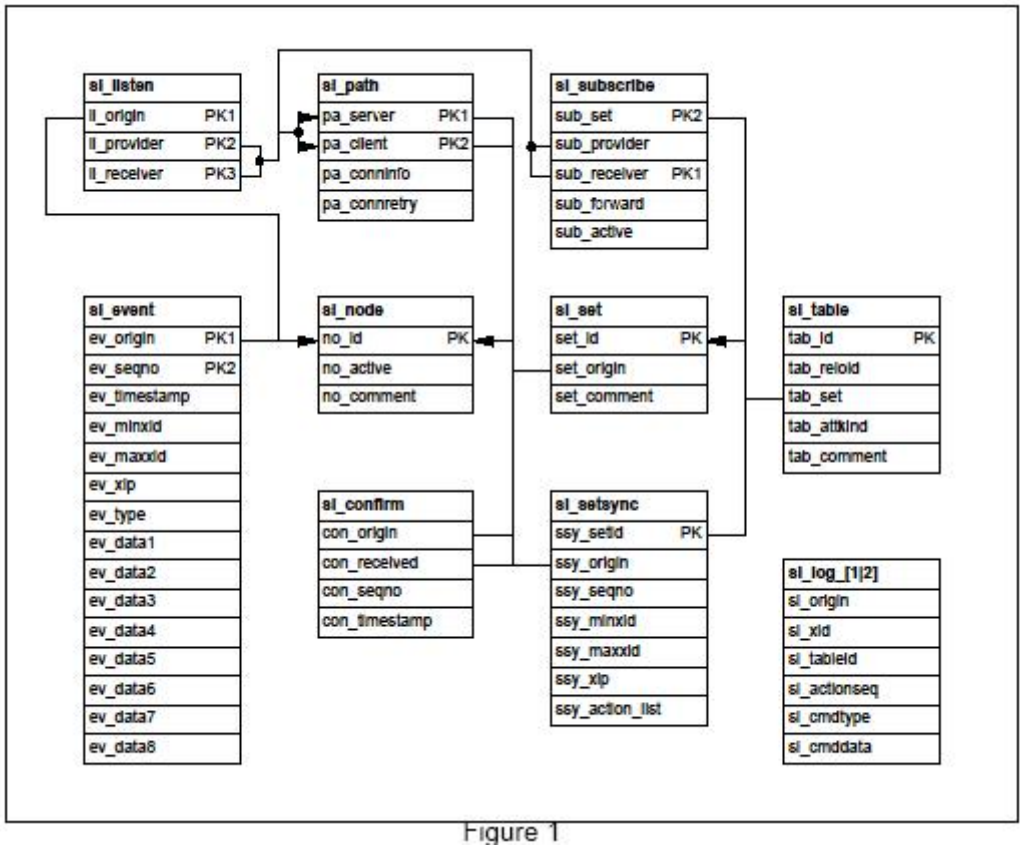

Figure 1

Figure 1 は Slony-I 構成と実行時データの構成要素関係図を示すものです。Slony-I はマスター・スレーブ複製技術とは言ってもクラスタを構成しているノードは何ら特定の役割を持っていません。全てのノードは同一の構成データを所有し、同一の複製エンジンプロセスを実行しています。いつの時点でもセットと呼ばれるテーブルを収集したものはそのオリジン（起源）として1つのノードを持っています。テーブルのオリジンは通常のクライアントアプリケーションにより更新が許可されている唯一のノードです。全てのノードが機能的には等しく、全般の構成データを共有している事実はフェイルオーバーとフェイルバックを非常に容易にします。全てのオブジェクトはクラスタ名にもとづく別々の名前空間に保存されます。

1.1. テーブル sl_node

クラスタに所属する全てのノードをリストします。属性 no_active は問題となっているノードでいかなる短期的な enable/disable 作業を目的としたものでは**ありません**。ノードの disable から enable への遷移はクラスタで完全な同期を必要とし、結果として全てのセットの複写操作を迫られる場合もあります。

1.2. テーブル sl_path

pa_client ノードが pa_server ノードに接続し、接続が失敗すれば秒単位の間隔で再試行するための接続情報を定義します。全てのノードがそれぞれお互いに接続できる必要はありません。しかし万一の場合のフェイルオーバーに備え構成が特定の場所にあるようになるため、全ての可能性のあるせつぞくを定義することは良い習慣です。sl_path エントリのみでは確立されるべき接続の要因とは実際になりません。接続には sl_listen エントリと（もしくは）sl_subscribe エントリも要求されます。

1.3. テーブル sl_listen

li_reciever ノードが選択し処理する、ノード li_provider へのデータベース接続を通し li_origin 上で発生する事象を指定します。古典的階層の通常のマスター・スレーブの筋書きでは事象は複製データと同じ経路に沿って伝わります。しかし異なったノードで複数のセットが発生する筋書きの場合、事象をより冗長に流通させる必要があります。

1.4. テーブル sl_set

セットは1つのノードで発生するテーブルとシーケンスの集まりで、他のどんなノードによっても購読される最小単位はクラスタです。

1.5. テーブル sl_table

テーブルとそれらのセットの関係をリストします。同時にログデータの更新情報を構築する複製トリガーの複製に使用されるテーブルの属性種類を指定します。

1.6. テーブル sl_subscribe

どのノードがどのデータセットに購読されるのか、およびどこからログデータを取得するのかを指定します。ノードはデータをセットオリジンもしくは発送（カスケード）で購読されるどんな他のノードからも受け取ることができます。

1.7. テーブル sl_event

これはメッセージを伝えるテーブルです。事象（構成の変更もしくはデータ同期）を生成するノードはこのテーブルに挿入される新規行を監視し、事象を監視している他の全てのノードに通知（Notify）します。事象を監視している遠隔（リモート）ノードはそこでそれらのレコードを select し、局所（ローカル）の構成もしくは複製データを変更し、自身のものとして局所の sl_event テーブルに sl_event 行を格納し、そこを通知（Notify）します。このようにして事象はクラスタ全般にわたってカスケード（上から下への情報伝達）します。SYNC 事象において、列 ev_minxid、ev_maxxid および ev_xip にはトランザクションの直列化可能スナップショット情報が書いてあります。これは PostgreSQL の MVCC で使用される情報と同一で、特定の変更が既にトランザクションに可視で

あるか、もしくは将来的に可視になると見なされるかどうかを告げます。Slony-I でデータは行レベルの単一の操作で複製されますが、2つの SYNC 事象の間で起こった全ての変更を含む1つのトランザクション内にグループ化されます。MVCC 可視化ルールにもとづいた最後で実際の SYNC 事象トランザクションを適用することはこのグループ化を行うためのフィルタ機構です。

1.8. テーブル sl_confirm

ノードで処理される全ての事象はこのテーブルで確認されます。確認は事象と似通ったシステムを通じてカスケードされます。複製エンジンの局所でのクリーンアップスレッドは定期的にこの情報を凝縮し、そして全てのノードで確認された sl_event 内の全てのエントリを削除します。

1.9. テーブル sl_setsync

このテーブルは全ての購読されたデータセットの現在の局所 sync 状態が何かを実際のノードにのみ告げます。この状態情報はシステム内の他のノードには複製されません。この情報は2つの目的のために使用されます。、最後の複製周期で可視にならなかったログ行を特定するためノードは複製の間トランザクションのスナップショットを使用します。データセットに新規購読された初期データ複写をノードが行う場合、どの sync 点と追加ログデータがこの実際のデータスナップショットに既に含まれているかの情報を使用します。

1.10 テーブル sl_log_1

このテーブルは複製トリガーでログが取得された実際の行レベルの変更を所有しています。データは頻繁に全てのノードが対応する事象を確認した後にクリーンアップスレッドにより削除されます。

1.11 テーブル sl_log_2

システムには sl_log_1 とこのテーブル間の切り替えを行う機能があります。クリーンアップスレッドが古くなったログ情報を削除し、開放区域マップに開放された区域を登録するため vacuum を使用し、通常環境においては同一のログテーブルをシステムに使用させるのが好ましいです。PostgreSQL は高度な同時性をもってよりよい並列化した挿入操作のため、開放区画マップで見つけられた複数ブロックを使用することが可能です。ノードがオフライン、もしくは他の理由により遥かに限界を超えて壊滅した場合、テーブルに収集されたログデータは容量がとてつもなく大きくなる可能性があります。このような場合、開放区間を取り戻すには full vacuum の実行しか方法がありませんが、これには排他的テーブルロックとアプリケーションの効果的停止が必要になります。これを回避するには、他のログテーブルにシステムを切り替え、古いログテーブルが論理的に空になった後、テーブルを空にします。

2. 複製エンジン基本構造設計

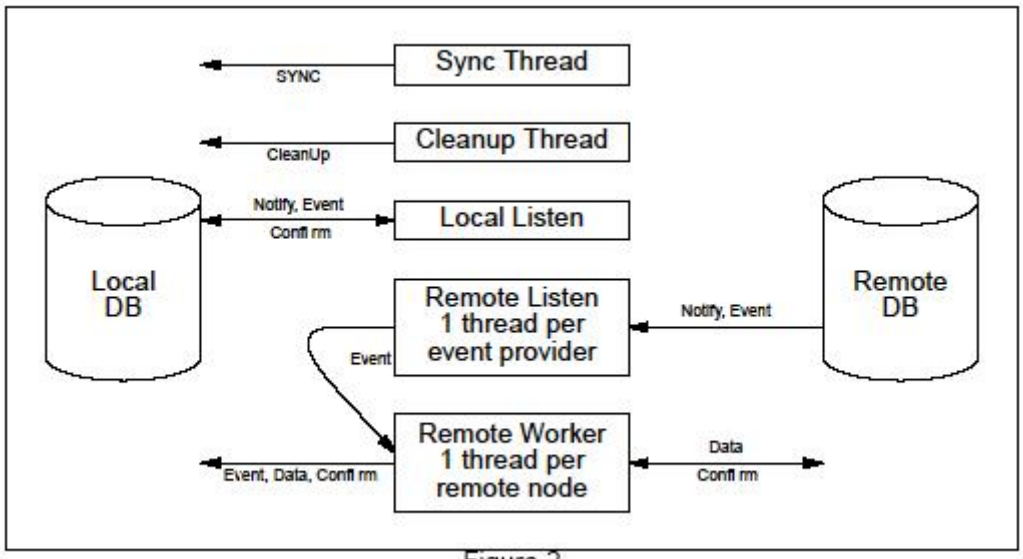

Figure 2

Figure 2 は Slony-I 複製エンジンのスレッド基本構造設計を描いています。Slony-I クラスタ内のいかなるノードも事前定義の役割は無いことを覚えておくことが重要です。従って、このエンジンはどんなクラスタのノードであってもよいデータベース毎に一回実行され、全てのエンジンは一緒に「1つの分散複製システム」を構築します。

2.1. Sync スレッド

Sync スレッドは局所データベースへの1つ接続を保守します。構成可能な時間間隔で、何らかの複製可能データベースが使用可能になったかを示す活動シーケンスの変更を検査します。そしてその後、CreateEvent() を呼び出しことにより SYNC 事象を生成します。他のスレッドとの相互作用はありません。

2.2. クリーンアップ（Cleanup）スレッド

クリーンアップスレッドは局所データベースへの1つの接続を保守します。構成可能な時間間隔で、古くなった確認済みの事象とログデータを削除する内部プロシージャ Cleanup() を呼び出します。別の時間間隔で確認済みの事象とログテーブルを vacuum します。他のスレッドとの相互作用はありません。

2.3. 局所監視（Local Listen）スレッド

局所監視スレッドは局所データベースへの1つの接続を保守します。「事象（Event）」通知に待機し、局所ノードで発生する事象の走査を行います。管理プログラムがクラスタ構成変更のため内部プロシージャを呼び出すことによる新規構成事象を受け取った場合は、複製エンジンンのメモリー内構成をそのように変更します。

2.4. 遠隔監視（Remote Listen）スレッド

遠隔ノード毎に1つの遠隔監視スレッドが存在し、局所ノードは（事象提供ノード：event provider）から事象を受け取ります。クラスタ内のノードの数に拘わらず同一の提供ノードを通じて全てのオリジンから事象を受け取るため、代表的な樹構造の葉ノードはたった1つの遠隔監視スレッドを所有します。遠隔監視スレッドは事象提供プロバイダへの1つのデータベース接続を保守します。事象の通知もしくは確認を受け取ると、対応するテーブルから新規情報を select し、それらを作業者（worker）スレッドの対応する内部メッセージ待ち行列に与えます。エンジンは遠隔ノード毎に1つの遠隔ノード特定の作業者スレッド（下記を参照）を開始します。メッセージは内部メッセージ待ち行列にもとづいて、処理と確認のためこのノード特定の作業者スレッドに転送されます。

2.5. 遠隔作業者（Remote Worker）スレッド

遠隔ノード毎に1つの遠隔作業者スレッドが存在します。遠隔作業者スレッドは、事象の保存と確認の実際のデータ複製アプリケーションを行う1つの局所データベース接続を保守します。作業者が処理している遠隔ノードで発生する全てのセットは1つのデータ提供ノード（在っても良いが、事象提供ノードと同じでは決してなりません）を持っています。これらのセットに渡り、異なったデータ提供ノード毎に、作業者スレッドは実際の複製データの選択を実行する1つのデータベース接続を保守します。遠隔作業者スレッドは遠隔作業者スレッド（複数可）により転送された事象にたいする内部メッセージ待ち行列上で待機します。そして、それらのデータの選択、アプリケーション、および確認を含む事象を処理します。同時にエンジンのメモリー内構成情報の保守も含まれます。