Vol.2014-DBS-160 No.5
Vol.2014-OS-131 No.5
Vol.2014-EMB-35 No.5
2014/11/18

ソーシャル SQL データベースを用いたアプリケーションの開発

李咏[†1]　新城靖[†1]　海沼直紀[†1]　潘進睿[†1]　佐藤聡[†1]　中井央[†1]

ソーシャル SQL データベースとは、各ユーザがそれぞれの計算機にデータベースを持ち、SNS(Social Networking Service)を利用して実現されたオーバレイネットワークにより相互利用が可能な関係データベースである。従来のソーシャル SQL データベースでは、一度に単一のデータベースを用いるアプリケーションしか開発されていなかった。ビューの切り替えによるデータベースのアクセス制御がやりにくかった。また、Web ブラウザ用の API は提供されていたが、それを活用したアプリケーションは存在しなかった。本研究では、ソーシャル SQL データベースを用いて複数のリモートデータベースを同時に利用できるアプリケーションを実現する。このアプリケーションを、Web ブラウザ用の API を用いて開発する。ローカルのデータベースと複数のリモートのデータベースを統合するために、本研究では MySQL の Federated エンジンを利用する。より簡便なアクセス制御を実現するために、本研究では MySQL のプロキシにおいて ACL(Access Control List)を評価し、SQL 文の置換、および、結果セットの書き換えを行う。本研究を評価するために、Twitter のタイムラインのようなアプリケーション、および、スケジュール管理アプリケーションを実装する。

1. はじめに

近年 SNS（Social Networking Service）をもとに動作する Web アプリケーションが開発されているが、その多くは中央サーバに依存する。そのような SNS は集中型 SNS であり、例としては、Facebook、Twitter、Google+が挙げられる。これらの集中型 SNS と連携したサービスやアプリケーションの開発も盛んである。

中央サーバに依存することにより、サービス停止に伴うデータ喪失及びプライバシーの侵害が起こることがある。例えば、Google Wave や Google Reader 等のサービスを終了するとともに、ユーザのプライベート情報が失われたことがある。2013 年には Edward Snowden 氏により、米国政府機関が集中型 SNS に対して大規模なプライバシー侵害行為を行っていることが告発された[1]。

このような集中型 SNS による問題を解消するため、分散型 SNS または分散型 OSN（Decentralized Online Social Networks）が提案されている[2][3]。分散型 SNS とは、中央サーバに依存せず、友達同士の繋がりや交流を行う機能を持つシャルアプリケーションを利用する仕組みである。ソーシャル VPN を用いて実装する分散型 SNS が存在する[4][5][6]。

分散型 SNS では、プライベート情報は全て個人のコンピュータや信頼できる連邦化されたサーバに保存される。そのため、集中型 SNS によるデータ喪失・漏洩及びプライバシー侵害の問題を解決できる。しかし、集中型 SNS の全ての機能を分散型 SNS で置き換えることができない。分散型 SNS では、公開鍵の交換等の形式による構築するため、堅牢性（durability）及び持続性を確保することができないことがある。例えば、ある友人からの公開鍵を紛失した場合には、友人を判明できなくなる。

パブリックな情報を集中型 SNS の中央サーバに蓄積し、友人関係の維持及びユーザ認証に利用する一方、プライベートな情報が分散型 SNS を通じて交換できるようにすると有益である。この時、集中型 SNS と分散型 SNS を連携させてと便利である[5]。

本研究室では、ソーシャル VPN を分散型 SNS の基盤とし、分散型 SNS におけるデータベースとしては、ソーシャル SQL データベース[7]を提案している。ソーシャル SQL データベースとは、各ユーザがそれぞれの計算機にデータベースを持ち、分散型 SNS を実現するために相互利用が可能な関係データベースである。しかしながら、従来のソーシャル SQL データベースには、次のような問題がある。

(1) 単一のローカルまたはリモートのデータベースを利用するソーシャルアプリケーションの開発が簡単であったが、複数のユーザのデータベースに同時に利用するソーシャルアプリケーションの開発が難しかった。
(2) ビューの切り替えによるアクセス制御では、予めユーザごとにアクセス権限に応じたビューを定義し、ユーザのアクセス権限の変更による新たなビューを定義必要があり、手間が大きかった。
(3) Web ブラウザ用の API は設計されたが、利用可能な SQL 操作が限定され、アプリケーションの開発に利用されなかった。

このような問題を解決するために、本研究では、次のようにソーシャル SQL データベース、および、そのアプリケーションを開発する。

(1) 各ユーザが設置したデータベースサーバを連邦化することにより、複数のユーザのデータベースに同時に利用することを可能にする。本研究では、MySQL の Federated エンジン[8]を利用して実現する。
(2) SQL クエリのインジェクション、および、置換によ

†1 筑波大学

Vol.2014-DBS-160 No.5
Vol.2014-OS-131 No.5
Vol.2014-EMB-35 No.5
2014/11/18

り、事前にユーザごとにアクセス権限を設定する必要をなくする。しかも、ユーザのアクセス権限の変更に動的に追随できるようにする。

(3) Web ブラウザ用の API を利用してアプリケーションを作成し、その機能と性能を評価する。

2. ソーシャル SQL データベースの実現するためのソーシャル VPN

本研究では、ソーシャル SQL データベースを、ソーシャル VPN を使用して実現する。ソーシャル SQL データベースの通信基盤としてソーシャル VPN は以下の要件を満足しなければならない。

(1) Socket API で TCP/IP による通信が可能であること。
(2) 認証済みのユーザ間では、暗号化されたチャネルを利用して通信を行うことができること。特に、ソーシャルアカウントを利用して認証を行えること。
(3) オンラインのユーザのリスト、および、各ユーザの端末の関連情報（ホスト名、IP アドレス等）を取得できること。
(4) 各ユーザの端末にソーシャルアカウントに関連付けたユニックな完全修飾ドメイン名（FQDN）を割り当てることができること。例えば、ソーシャルアカウント Bob を利用して端末の完全修飾ドメイン名は pc1.bob.social-vpn である。

この要件を満たすソーシャル VPN は次のものがある。

- SocialRouter[4]
- Social SoftEtherVPN[5]
- SocialVPN[6]

本研究では、Social SoftEtherVPN を利用して実装する。

3. データベースの統合

従来のソーシャル SQL データベースには、クライアントが複数のデータベースに同時に利用することを支援していない。そのため、クライアント側にはそれぞれのデータベースから返された結果を自ら処理する必要があり、アプリケーション開発の負担が大きかった。

この問題を解決するため、本研究では分散されたデータベースを仮想的に統合するフェデレーショ技術を用いる。これにより、複数の SNS メンバーのデータの統合を可能にする。統合サーバとして稼働するデータベースには、分散されたデータベースの表を関連付けた仮想的な表（ニックネーム）が登録できる。SQL 文でニックネームを表のように使用することで、分散されたデータは、あたかも統合サーバ上のデータベースに存在するかのように扱えることができる。

ニックネームを参照する SQL 文が統合サーバに対して発行されると、統合サーバは SQL 文に含まれた各構文をローカル上で実行するか、リモートのデータベースで実行するかを決定する。リモートのデータベースへ用の SQL 文を作成しリモートのデータベースに処理を要求する。INSERT、UPDATE、および、DELETE 文の場合は処理結果を、SELECT 文の場合は処理結果に加え結果セットをリモートのデータベースから受け取り、統合サーバ上で必要な処理を実行したのち、クライアントに結果を返す[9]。

図 1 の例では、3 つのリモートのデータベース Bob, Carol, Dave を統合し、リモート表 TB, TC, TD をそれぞれのニックネーム NB, NC, ND に関連付けている。この構成では、ソーシャルアプリケーションから TA, NB, NC, ND を参照する SQL 文が統合サーバ Alice に発行されると、ローカル（Alice）及びリモートのデータベース(Bob, Carol, Dave)に対して適切な SQL 文を発行し、それらの結果から統合サーバに対して発行された SQL 文の結果セットを求めてソーシャルアプリケーションに返す。

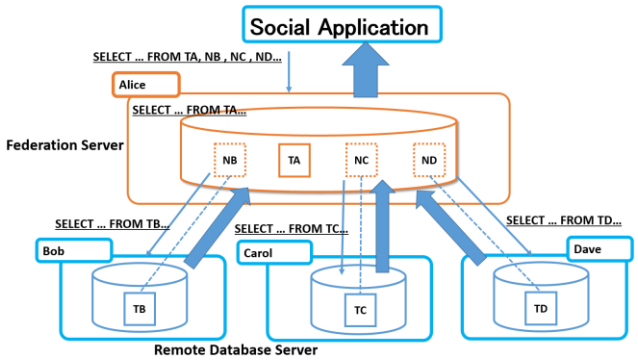

図 1 分散されたデータベースのフェデレーションによるソーシャル SQL データベースの構成

本研究では、MySQL の Federated エンジンを利用することで、複数の SNS メンバーのデータベースの統合を実現する。Federated エンジンとは、MySQL5.0 から提供する分散されたデータベースを仮想的に統合させる機能である。

統合サーバに稼働する MySQL データベース（以下、MySQL DB と呼び）にリモート MySQL DB の表と同じのスキーマで関連付けるフェデレーション表を生成することができる。統合サーバの MySQL DB にリモートの MySQL DB の表を関連付けたフェデレーション表を作成する例を図 3 に示す。

Vol.2014-DBS-160 No.5
Vol.2014-OS-131 No.5
Vol.2014-EMB-35 No.5
2014/11/18

```
1   CREATE TABLE `Tweets` (
2       `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
3       `text` varchar(160) NOT NULL,
4       `text` varchar(160) NOT NULL,
5       `category_id` int(4) DEFAULT NULL,
6       `created_at` datetime NOT NULL,
7       `lang` varchar(10) DEFAULT NULL,
8       `retweet_count` bigint(10) DEFAULT '0',
9       `retweeted` tinyint(1) DEFAULT '0',
10      `truncated` tinyint(1) DEFAULT '0',
11      PRIMARY KEY (`id`)
12    (ENGINE=MyISAM AUTO_INCREMENT=107 DEFAULT CHARSET=utf8
```

図 2 Bob 上の Tweets 表のスキーマ

```
1  CREATE TABLE `Tweets_Bob` (
2       `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
3       `text` varchar(160) NOT NULL,
4       `text` varchar(160) NOT NULL,
5       `category_id` int(4) DEFAULT NULL,
6       `created_at` datetime NOT NULL,
7       `lang` varchar(10) DEFAULT NULL,
8       `retweet_count` bigint(10) DEFAULT '0',
9       `retweeted` tinyint(1) DEFAULT '0',
10      `truncated` tinyint(1) DEFAULT '0',
11      PRIMARY KEY (`id`)
12 (ENGINE=FEDERATED DEFAULT CHARSET=utf8
13 CONNECTION='mysql://alice@pc1.bob.social-vpn:3306/twitter/Tweets'
```

図 3 統合サーバ Alice に Bob の Tweets と関連付けたフェデレーション表のスキーマ

図 2 ではリモートサーバ(Bob)上の MySQL DB の Tweets 表のスキーマを表示する。Bob のホスト名は pc1.bob.social-vpn であり、ポート番号は 3306 である。

図 3 の 1 2 行目は Federated エンジンを指定して、CONNECTION を Bob に稼働している twitter データベースの Tweets 表が参照できる URL にする。図 3 の 2 から 1 1 行目までは統合サーバ（Alice）には、Bob 上の Tweets 表と同じのスキーマで関連付けたフェデレーション表（twitter.Tweets_Bob）を作成している。

4. データベースのアクセス制御

本研究では、ソーシャル SQL データベースを、MySQL を用いて実現する。MySQL Proxy は、MySQL クライアントとサーバの通信を中継するソフトウェアであり、軽量なプログラミング言語 Lua を用いて中継する要求や応答を書き換えることができる。本ソーシャル SQL データベースはこれらの特性を利用してアクセス制御を実現する。

4.1 従来のソーシャル SQL データベースのアクセス制御

従来のソーシャル SQL データベースでは、ビューを用いたアクセス制御が行われていた[7]。この方法では、既存の属性などを利用して事前にいくつかのビューを生成する。そして、生成したビューにそれぞれユーザを対応させる。ソーシャル SQL データベースでは、SNS ソーシャル VPN よりリモートユーザの SNS アカウントを得られる。したがって、「ユーザとビューの対応」を参照してビューの切り替えにより相応しいビューを返す。

この方法の問題点としては、事前に各ユーザに応じて複数のテーブルに対し結合演算、選択演算及び射影演算を行い作成したビューを用意しなければならないことである。また、ユーザの信頼の度合いの変更に伴いビューを再定義しなければならないという問題もある。

4.2 提案するソーシャル SQL データベースのアクセス制御

本研究では、事前のビュー定義を行うことなくアクセス元の SNS アカウントに応じてデータベースで表、および、行レベルのアクセス制御を行う。データの持ち主であるユーザは、自分のデータベースサーバに接続してくるユーザごとに、アクセス用のユーザアカウントを作成する。それぞれのアカウントに応じてアクセスできる資源を、ACL（Access Control List）を利用して設定する。

未登録のユーザアカウントでアクセスした際に、アクセスを拒否する。登録済みの場合は、ユーザアカウントに応じて限定される資源のみアクセス可能にする。

本研究では、MySQL プロキシ[10]を利用して SQL クエリインジェクション、および、SQL クエリ置換２つのアクセス制御手法を提案する。SQL クエリインジェクションによるアクセス制御は、リモートからのクエリとローカルサーバ返した結果を書き換える。一方、SQL クエリ置換によるアクセス制御は、リモートからのクエリのみ書きかえる。

これらのアクセス制御実現手法では、両方とも共通の ACL を用いる。以下の項では、まずその ACL の実装方法について述べる。次に、SQL クエリインジェクションによる手法、および、SQL クエリ置換による手法について述べる。

4.3 アクセス制御のための ACL 関連表

本研究では、次の３つの表を用いて ACL を表現する。

- ユーザ表
- リソース表(Resource Table)
- アクセス権表(Access privileges Table)

このうち、ユーザ表とリソース表は、それぞれ木構造で表現している。これらの３つの表をまとめて、**ACL 関連表**と呼ぶことにする。

Vol.2014-DBS-160 No.5
Vol.2014-OS-131 No.5
Vol.2014-EMB-35 No.5
2014/11/18

ユーザ表は SNS のアカウント名を用いて MySQL アクセス用のユーザを定義する。また、SNS のアカウント名と MySQL アクセス用のユーザの間に 1 対 1 関係を持っている。

リソース表はアクセスを許すリソースを定義する。本研究では、リソース表では表、および、レコード集合を単位とし管理する。

アクセス権表はユーザとリソースを関連づける表である。

図 4 と図 5 では、アプリケーションデータ用の 3 つの表(Table A, Table B, Table C)と、それへのアクセス制御のポリシーを概念的に記述したアクセス行列である。ユーザ Alice は、Table A に対して Insert, Select, Update, および Delete 権限を、Table B のレコード集合である b1, b2 に対して Select 権限を所有している。ユーザ Bob は、Table A に対して Select 権限を、b1, b2 に対して Select, Insert 権限を所有している。ユーザ Carol は、Table A に対して Select 権限を、Table C のレコード集合である c2 に対して Insert, Select, Update, Delete 権限を所有している。

図 6 は、図 5 のアクセス行列で表現したポリシーを、ACL 関連表(ユーザ表を 省略)で実現したものを示している。リソース表では、Table A をリソース TA とし、Table B のレコード集合である b1, b2 をリソース TB_Rs1 とし、Table C のレコード集合である c2 をリソース TC_Rs2 とし定義している。リソース表は、アクセス権表と結合して、ポリシーを表現している。図 6 では、アクセス権限表からユーザ Alice, Bob, Carol はそれぞれアクセスできるデータをアプリケーション用のデータ表までたどり着ける。図 6 の ACL 関連表は、図 5 に示したアクセス行列と同じアクセス制御のポリシーを実現している。

Field
a1
a2
a3

Table A

Field
b1
b2
b3

Table B

Field
c1
c2
c3

Table C

図 4 アプリケーションデータの A, B, C 表

User	TA	TB_Rs1(b1,b2)	TC_Rs2(c2)
Alice	*Insert,Select, Update,Delete*	Select	
Bob	Select	*Select,Insert*	
Carol	Select		Insert,Select, Update,Delete

Access Matrix

図 5 アクセス行列

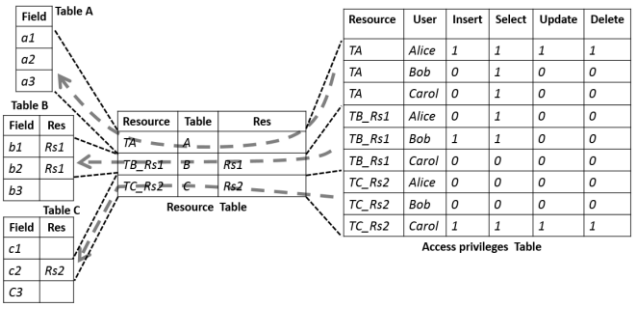

図 6 ACL 関連表（ユーザ表を省略）

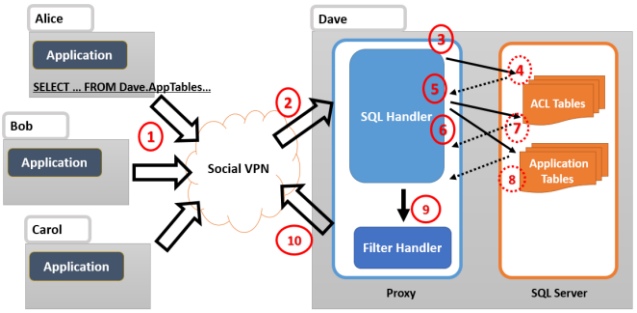

図 7 SQL クエリインジェクションによる
ソーシャル SQL データベースへのアクセス制御

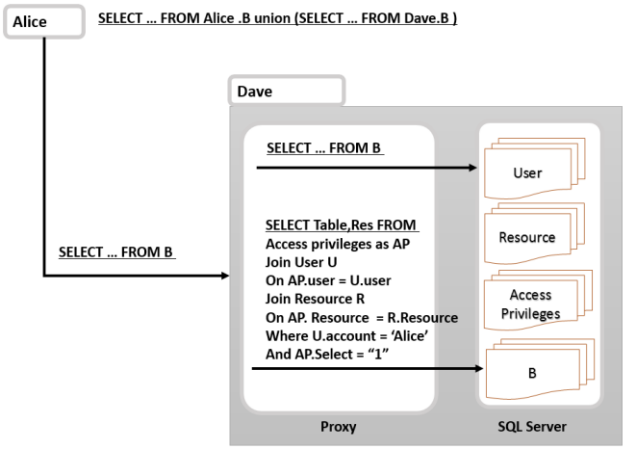

図 8 SQL クエリインジェクションによる新たな SQL クエリの生成

4.4 SQL クエリインジェクションによるソーシャル SQL

Vol.2014-DBS-160 No.5
Vol.2014-OS-131 No.5
Vol.2014-EMB-35 No.5
2014/11/18

データベースのアクセス制御

この節では、SQL クエリインジェクションによるアクセス制御の実装について述べる。この方法では、リモートからのクエリとローカルサーバが返した結果を、4.2 節で述べた ACL 関連表を参照するように書き換える。この書き換えを、図 7 を用いて説明する。

図 7 では、リモートユーザである Alice、Bob、Carol がソーシャル VPN を経由してユーザ Dave のデータベースにアクセスしている。リモートユーザ Alice が Dave 側に **SELECT** 操作のアクセスする場合、以下の処理が行われる。

(1) Alice 側では、ソーシャル VPN による作った仮想トンネルを通り、TCP/IP プロトコルを利用して指定したポート番号で Dave 側に稼働している MySQL プロキシサーバに接続する（図 7 の 1）。

(2) Dave 側のプロキシサーバは、Alice からの接続要求を受け付ける。SQL ハンドラ（SQL Handler）が SQL クエリからプライベート IP アドレスを得られる。そして、この IP アドレス用いてソーシャル VPN に接続元を照会して、SNS の友人であれば、該当するユニックなソーシャルドメイン（FQDN）を得、信頼できる通信路を作成する（図 7 の 2）。

(3) SNS の友人ではない場合、アクセスを拒否し、該当するエラーを返す。

(4) Dave のプロキシサーバは、SQL ハンドラが受け取った SQL クエリが正規表現でマッチしなかった、つまりアプリケーション用のデータ表（Application Tables）以外にアクセスした場合には、アクセスを拒否し、該当する SQL エラーを返す。

(5) SQL クエリが正規表現でマッチした場合のみアクセスを許可する。

(6) (2)で取得したソーシャルドメイン名からデータベースに登録したカウント名(Alice)を得る。そして、Alice のアカウント名を利用してデータベース上の ACL 関連（ACL Tables）のユーザ表を参照する SQL クエリを生成して、データベースサーバに送る（図 7 の 3）。

(7) データベースサーバが実行された結果を SQL ハンドラに返す（図 7 の 4）。

(8) Dave 側に Alice が予め登録されなかった場合には、該当する SQL エラーを返す。

(9) Dave 側に Alice が予め登録された場合には、Alice のアカウント名を利用して ACL 関連表を参照する SQL クエリを生成する。Alice からの元の SQL クエリとともに、データベースサーバに送る（図の 5，6）。

(10) 2 つの SQL クエリがデータベースに実行されたのち、2 つの結果セットを SQL ハンドラに返す（図 7 の 7，8）。

(11) SQL ハンドラは、受け取った 2 つ結果セットをフィルタハンドラ（Filter Handler）に渡す（図 7 の 9）。

(12) フィルタハンドラでは、返した結果を Alice の SELECT 操作のアクセス権限に基づいてフィルタリングを行う。処理済みの結果を Alice に返す（図 7 の 10）。

上の(6)では、4.3 節で述べた ACL 関連表を参照するような SQL クエリを生成する。生成されるクエリの例を、図 8 に示す。

この方法の利点は、事前に各ユーザに応じてビューを用意する必要がないことである。また、アクセス権限の変更はただ ACL 関連表の更新によって実現できる。

この方法の問題点は 3 つがある。

1 つ目は、SELECT 文の SQL クエリのみ対応していることである。INSERT、 UPDATE、および、DELETE 文の場合は、ACL 関連表を参照する SQL 文とは別々実行されるため、アクセス制御をすることができない。 2 つ目は、アプリケーション用の表におけるレコード件数が増えると伴い、結果のフィルタリングが重たくなり、レスポンスタイムがより長くなることである。 3 つ目は、データベースのスキーマが変更すれば、フィルタハンドラにおける処理が新たなスキーマに沿って修正する必要があることである。

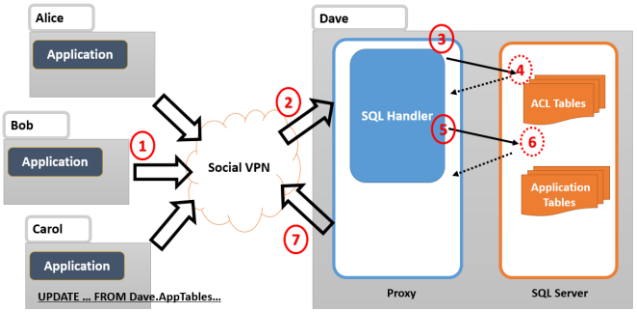

図 9 SQL クエリ置換による
ソーシャル SQL データベースへのアクセス制御

4.5 SQL クエリ置換によるソーシャル SQL データベースのアクセス制御

この節では、SQL クエリ置換によるアクセス制御の実装について述べる。この方法では、リモートからのクエリのみを ACL 関連表を参照するように書き換える。この書き換えを、図 9 を用いて説明する。

図 9 では、リモートユーザである Alice、Bob、Carol がソーシャル VPN を経由してユーザ Dave のデータベースにアクセスしている。

ここでは、リモートユーザ Carol が Dave 側に **UPDATE** 操作のアクセスをしたとする。図 9 の 1 から 4 までの処理は、4.4 節に述べた処理（図 7 の 1 から 4 まで）と同じである。図 9 の 5 以降を説明する。

Vol.2014-DBS-160 No.5
Vol.2014-OS-131 No.5
Vol.2014-EMB-35 No.5
2014/11/18

(1) Dave 側に Carol が予め登録された場合には、Carol からの SQL クエリを元に、ACL 関連表と結合して UPDATE 権限を所有しているデータのみ対象とし、新たな SQL クエリを生成する。生成した SQL クエリをデータベースサーバに送る（図９の５）。

(2) SQL クエリがデータベースで実行されると、結果を SQL ハンドラに返す（図９の６）。

(3) SQL ハンドラは返した結果を処理せず、Carol に返す（図９の７）。

4.4 節で述べた SQL クエリインジェクションによるアクセス制御では、1 回のアクセスにつき SQL クエリが 3 回実行される方法と比較して、この方法の利点は、データベースから返した結果のフィルタリングというステップがないこと、および、1 回のアクセスにつき SQL クエリが 2 回だけ実行されるので、レスポンスタイムがより短いことである。

5. 分散型 Web ブラウザ上でのソーシャルアプリケーションの実装

分散型 Web ブラウザ[11]とは、本研究で開発している Web ベースのソーシャルアプリケーションの実行環境である。分散型 Web ブラウザには、Java Applet を利用してソーシャル SQL データベースを利用する機能がある。本研究では、この機能を利用して Twitter のタイムラインのようなアプリケーション、および、スケジュール管理アプリケーションを実装する。

5.1 分散型ブラウザの API の利用

分散型 Web ブラウザは、HTML5 を参考にして設計された、データベースをアクセスするための API を持っている。HTML5 では、データベース関連 API として Web SQL Database、および、Indexed Databased API が提案されている。Web SQL Database は Web ブラウザ上で SQLite による関係データベースを動作させる API であり、データベースの各種の操作を SQL 記述できる。Web SQL Database はそもそも HTML5 標準仕様として検討されてきて 2010 年に W3C では Recommendation に入れるのをやめた。その代わりに Indexed Database API を利用することが推奨されている。

この２つの API はクライアント側のデータベースに限って操作できて TCP/IP でリモートのデータベースを操作するのは不可能である。

本研究では、分散ブラウザの API を利用してウェブアプリケーションを記述する。この API は以下の特徴を持っている。

- イベント駆動である。データベース接続またはクエリが完了した際に、実行される処理をコールバック関数で指定する。
- マルチスレッド処理で、データベースとの問い合わせは非同期で実行される。

これらにより、複数のデータベースを同時に操作するのが可能である。表 1 には、この JavaScript API の主な関数を挙げられる。

表 1　JavaScript API の説明

関数	説明
Client(host, database, user, password, port);	ローカルまたはリモートのデータベースを操作するオブジェクトを返す。
client.connect(callback);	指定したデータベースに接続する。コールバック関数には、接続成功時は null が、失敗時は例外送出可能なエラーオブジェクトが渡される。
client.query(sql,option,callback);	非同期で SQL を実行する。引数は SQL 文、バインド変数、コールバック関数である。バインド変数とコールバック関数はそれぞれ省略可能である。 コールバック関数の引数は、SQL が SELECT の場合は function(err,results,fields) となり、SELECT 以外の場合は function(err,fields)である。

5.2 Twitter タイムラインのようなアプリケーション

クライアント側では、5.1 節述べた JavaScript API を利用して実装した。

図１０では、複数の SNS メンバーの Tweet を検索することを示す。５行目では、まず、オンラインのソーシャル SQL データベースのリストを取り出す。17 行目では、普通の SQL の Join 操作で複数のソーシャル SQL データベースの表を結合して複数の SNS メンバーが投稿した Tweet 文を取得できる。

6. 関連研究

分散型 SNS の実装方法としては、主に次の３種類に分類される。

(1) サーバを連邦化（federation）するもの
ユーザは信頼する組織が設置したサーバ、または個人で設置したサーバにデータを保存する。例としては、Diaspora[12]、OneSocialWeb[13]、Persona[14]が挙げられる。

(2) P2P（Peer-to-Peer）ネットワークや DHT（Distributed Hash Table）を利用するもの
個人のコンピュータで、SNS アプリケーションを動作させる。例としては、PeerSoN[15]、Safebook[16]、LifeSocial[17]が挙げられる。

Vol.2014-DBS-160 No.5
Vol.2014-OS-131 No.5
Vol.2014-EMB-35 No.5
2014/11/18

(3) ソーシャル VPN を使うもの
ソーシャルアカウント名を利用して認証を行い、ユーザ同士は VPN 環境で繋がっている。
例としては、２章に述べた SocialVPN、Social SoftEtherVPN、SocialRouter が挙げられる。

(1)、(2) と比較して、(3) 利点は、ソーシャル SQL データベースを用いてソーシャルアプリケーションの開発環境の構築ができることである。

HomeViews[11]は、peer-to-peer により、データベースを共有する仕組みであり、アクセス制御にビューを用いている。HomeViews では、共有されるビューに対してケーパビリティに基づくアクセス制御を行う。これに対して、本研究では、ACL を用いてアクセス制御を行っている点が異なる。また、本研究では、ソーシャル VPN の通信路を利用するため、Facebook、Twitter 等の SNS を連携できるという点で優れている。

```
1 var client = new Client(); // connect to localhost database
2 client.connect(function(err){
3   if(err) throw err;
4   /* fetch online remote database from Tweet_servers */
5   client.doAsynQuery("select * from Tweet_servers", "getServersInfo");
6 });
7 });

8 var getServersInfo = function(identity, err,results,fields){
9 var errObj;
10 var resultsObj;
11 var fieldsObj;
12 if(err) errObj = $j.parseJSON(err);
13 if(results) resultsObj = $j.parseJSON(results);
14 if(fields) fieldsObj = $j.parseJSON(fields);
15 if(errObj) throw err;
16 $j.each(resultsObj, function(k, value){
17     querySql += "union ( select id, text, created_at , ¥'"+this['Server_name']+"("+this['Host']+")"+"¥' as local_or_remote from "+value['Server_name'] +" )";
18 });
19 querySql += " order by created_at desc limit ? ";
20 client.doAsynQuery(querySql,[['Int',10]], "render");
21}
```

図１０クライアントで Tweet 文を取得する JavaScript ソースコード

7. おわりに

本研究では、ソーシャル SQL データベースを用いて複数のリモートデータベースを同時に相互利用できるアプリケーションを実現する。現在までに、Twitter のタイムラインのようなアプリケーションの実装を完成した。今後は、実装したアプリケーションを用いて本研究で実現したアクセス制御手法を評価する。そして、Social SoftEtherVPN 以外の VPN も利用可能にしたいと考えている。

参考文献

[1] Greenwald, G. No place to hide. Edward Snowden, the NSA, and the US Surveillance state. Signal(2014).

[2] Datta, A., Buchegger, S., Vu, L. H., Strufe, T., & Rzadca, K. Decentralized online social networks. In Handbook of Social Network Technologies and Applications (pp. 349-378). Springer US (2010).

[3] Schwittmann, L., Wander, M., Boelmann, C., & Weis, T. Privacy Preservation in Decentralized Online Social Networks. IEEE Internet Computing, 1(2013).

[4] 櫻井孝一, 海沼直紀, 新城靖, 佐藤聡, 須藤侑一, 肖焜瑤, 中井央. 安全な家庭向けソーシャルルータの実現. 情報処理学会システムソフトウェアとオペレーティングシステム研究会 2013-OS-127, 2013

[5] 海沼, 新城靖, 登, 肖, 佐藤聡, 中央: "プライバシを保護するための VPN を用いた ソーシャルアプリケーション実行環境", 情報処理学会コンピュータシステム・シンポジウム,12 ページ (2014).

[6] Socialvpn : http://socialvpn.org/.2014-10-10.

[7] 須藤侑一, 新城靖, 櫻井孝一, 佐藤聡, 肖焜瑤, 中井央. ソーシャルネットワークを利用した SQL データベースの相互利用. 情報処理学会システムソフトウェアとオペレーティングシステム研究会 2013-OS-127, 2013.

[8] MySQL 5.6 Reference Manual. MySQL Federated, 2012. http://dev.mysql.com/doc/refman/.

[9] Sheth, A. P., & Larson, J. A. Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys , 22(3), 183-236(1990).

[10] MySQL 5.6 Reference Manual. MySQL Proxy, 2012. http://dev.mysql.com/doc/refman/.

[11] Shinjo, Y., Guo, F., Kaneko, N., Matsuyama, T., Taniuchi, T. and Sato, A.: A distributed web browser as a platform for running collaborative applications, 2011 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), pp. 278-286 (2011).

[12] A. Bielenberg, L. Helm, A. Gentilucci, D. Stefanescu, and H. Zhang, "The growth of Diaspora - A decentralized online social network in the wild", ;in Proc. INFOCOM Workshops, pp.13-18(2012).

[13] Sahama T, Liang J, Iannella R. Impact of the social networking applications for health information management for patients and physicians, Pro-ceedings of the 24th European Medical Informatics Conference(MIE2012), Vol. 180, pp. 803-807(2012).

[14] Baden, R., Bender, A., Spring, N., Bhattacharjee, B., & Starin, D. Persona: an online social network with user-defined privacy. ACM SIGCOMM Computer Communication Review, 39(4), 135-146(2009).

[15] Buchegger, S., Schiöberg, D., Vu, L. H., & Datta, A. PeerSoN: P2P social networking: early experiences and insights. In Proceedings of the Second ACM EuroSys Workshop on Social Network Systems (pp. 46-52). ACM(2009).

[16] Cutillo, L. A., Molva, R., & Strufe, T. Safebook: A privacy-preserving online social network leveraging on real-life trust. Communications Magazine, IEEE, 47(12), 94-101(2009).

[17] Graffi, K., Gross, C., Stingl, D., Hartung, D., Kovacevic, A., & Steinmetz, R. LifeSocial. KOM: A secure and P2P-based solution for online social networks. In Consumer Communications and Networking Conference (CCNC), (pp. 554-558). IEEE(2011).

[18] Geambasu Roxana, Balazinska Magdalena, Steven D. Gribble, and Henry M. Levy. Homeviews: peer-to-peer middleware for personal data sharing applications. In Proceedings of the 2007 ACMSIGMOD international conference on management of data, pp. 235–246, 2007.