Vol.2013-OS-127 No.1
Vol.2013-EMB-31 No.1
2013/12/3

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

須藤 侑一[1]　新城 靖[1]　櫻井 孝一[1]　佐藤 聡[1]　肖 焜瑤[1]　中井 央[1]

概要：本研究では、中央サーバを用いることで生じるプライバシ侵害の問題を解決するため、ソーシャル SQL データベースを提案する。ソーシャル SQL データベースとは、各ユーザがそれぞれ自身の計算機にデータベースを持ち、それを相互に利用することで通信に中央サーバを介さず、各ユーザ間で関係データベースの相互利用を可能にする。本研究では、ユーザの認証に SNS のアカウントを用いて、プロセス間通信にソーシャルルータが提供する安全な通信路を用いる。アクセス制御には、関係データベースのビューの機能を用いる。SNS アカウントに応じて、ユーザが利用できるビューを切り替えることで柔軟なアクセス制御を実現する。具体的なデータベースとして MySQL を利用し、ビューを切り替えるために MySQL のプロキシを利用する。本研究ではさらに Web ブラウザ上で動作する協調アプリケーションが、ソーシャル SQL データベースを利用するための JavaScript API を提供する。本研究を評価するために、ブログ WordPress および簡単な写真共有アプリケーションとソーシャル SQL データベースを連携させる。

1. はじめに

近年、SNS (Social Network Service) 上で動作する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。中央サーバに依存することによりスケーラビリティが低いという問題、サービス終了に伴う保存データの喪失、プライバシの侵害などが起こる可能性がある [1]。2013 年 6 月には Microsoft 社、Facebook 社などからアメリカ政府当局がプライバシを侵害するかたちで個人情報を収集していた事実が判明し、問題になっている。SNS のアプリケーションでは、データの保存などにおいてデータベースが使われている。データベースを中央サーバに依存しない形で利用できれば先に述べたような問題を解決することができる。

本研究では、このような問題を解決するために**ソーシャル SQL データベース**を提案する。ソーシャル SQL データベースは SNS のユーザ間で相互に関係モデルに基づき SQL によりアクセスできるデータベースを提供する。また SNS のユーザ名を使ってアクセス制御を行うことができる。

2. ソーシャル SQL データベース

ソーシャル SQL データベースは各ユーザが自身の計算機に持つデータベースをリモートの別のユーザにアクセスさせる。データの保存には関係データベースを用いる。サーバとクライアント間の通信はソーシャルルータ [2] が提供する通信路を用いる。ソーシャルルータとは SNS 用いてユーザ認証を行うことで他の SNS のユーザの計算機がルータの向こう側に存在するように通信できる仕組みであり、本研究室において開発を行なっている。

2.1 共有モデル

ソーシャル SQL データベースは各ユーザが自身の計算機に持つデータベースを相互に利用するため、通信に中央サーバを介さず各ユーザ間で直接通信する。図 1 に現在 SNS で多く利用されている各ユーザとデータベースの関係を示す。この図で、SQL クライアントは、多くの場合、Web アプリケーションである。ユーザのデータは中央のサーバに保存されるため、中央サーバが監視されると全ユーザの個人情報が収集されてしまう。図 2 に本研究の各ユーザとそのデータベースの関係を示す。各ユーザが中央のサーバを介すことなく自身の計算機に持つデータベースを直接相互利用する。

2.2 ソーシャル SQL データベースのアプリケーションのモデル

ソーシャル SQL データベースを使用するアプリケーションの 3 つのモデルを以下に示す。

(1) 非 Web アプリケーション

[1] 筑波大学

(2) Web アプリケーションつまり Web サーバ上で動作するアプリケーション
(3) Web ブラウザ上で動作するアプリケーション

2.2.1 非 Web アプリケーション

非 Web アプリケーションは、図 3 のように Web を介さないアプリケーションで SQL クライアントと SQL サーバのみで構成される。クライアントは直接サーバに通信を行い、データベースにアクセスする。例としては、カレンダーなどのスケジュール管理アプリケーション、掲示板アプリケーションが挙げられる。

2.2.2 Web アプリケーション

Web アプリケーションは、図 4 のように Web を介するアプリケーションで Web クライアント、Web サーバ及び SQL サーバにより構成される。図 4 のように Web アプリケーションは自分のデータベースまたはリモートのデータベースにアクセスする。データベースを使用する Web アプリケーションは SQL クライアントとしてサーバにデータを要求する。例としては、Apache 上のブログアプリケーションや写真共有アプリケーションである。

2.2.3 Web ブラウザ上で動作するアプリケーション

Web ブラウザ上で動作するアプリケーションは、図 5 のように Web ブラウザと SQL サーバにより構成される。Web ブラウザは自分のデータベースまたはリモートのデータベースにアクセスする。Web ブラウザ上のアプリケーションはデータベースに接続する。Web ブラウザ上で動作するアプリケーションは多くの場合、JavaScript によって記述される。例としては、分散型 Web ブラウザ [3] で動作する Web アノテーションが挙げられる。**分散型 Web ブラウザ**[3] とは、Web 上で協調アプリケーションを実行する基盤である。その特徴は中央サーバを介さずに分散型 Web ブラウザ間で直接通信を行うため、中央サーバに依存する問題を解決できることである。

本研究では、分散型 Web ブラウザのアプリケーションに対して関係データベースを利用可能にする。従来の分散型 Web ブラウザ上で動作するアプリケーションは各ユーザの計算機または SkypeFS[3] にファイルを保存することができる。SkypeFS は Skype を利用してクライアントからファイル単位でデータを読み書きする仕組みである。また従来の分散型 Web ブラウザは key-value 形式でデータを保存することができる [3]。これは Linda タイプのタプルスペースのタプル単位でアクセス制御を行なっている。本研究で実現するソーシャル SQL データベースにより、分散型 Web ブラウザのアプリケーションでも関係データベースを利用可能になる。

2.3 SNS を利用した通信路

ソーシャル SQL データベースは各ユーザの計算機間で中央サーバを介することなく通信するために SNS を利用した通信路を用いる。またユーザ認証を SNS サーバで行うことができる。ソーシャル SQL データベースで SNS を利用した通信路を構成するためにはソーシャルルータを用いる。

ソーシャルルータ [2] とは、家庭用ルータを拡張したもの

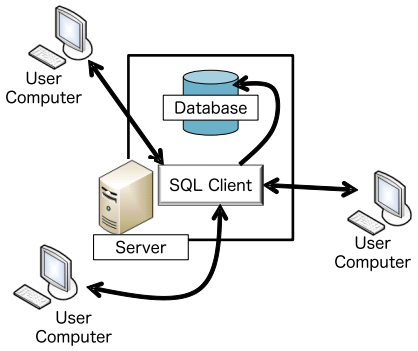

図 1 中央サーバを介したユーザの通信

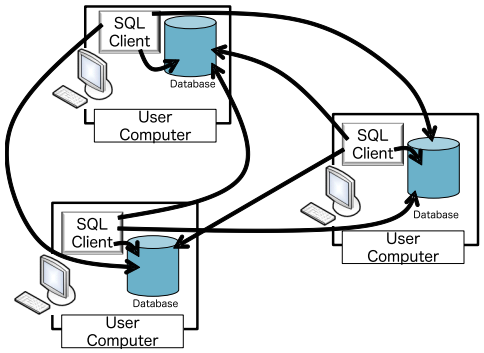

図 2 ユーザ間のデータベースの相互利用と直接通信

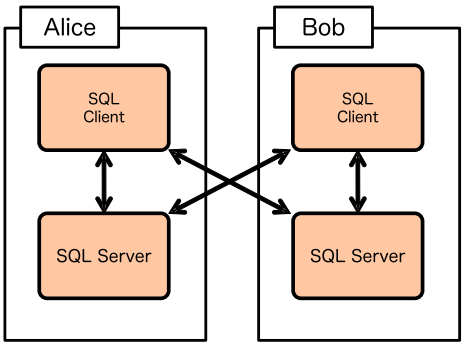

図 3 非 Web アプリケーション

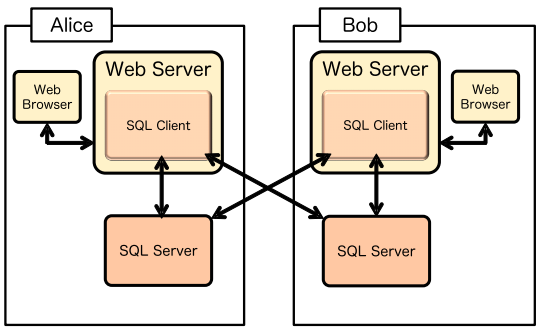

図 4 Web アプリケーション

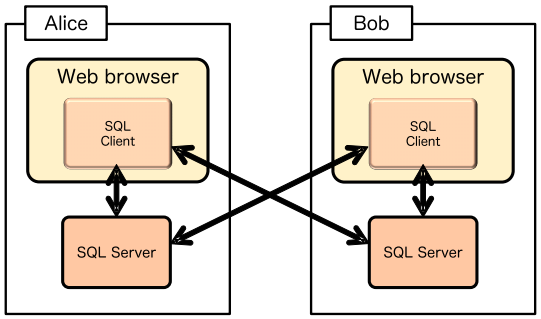

図 5 Web ブラウザ上で動作するアプリケーション

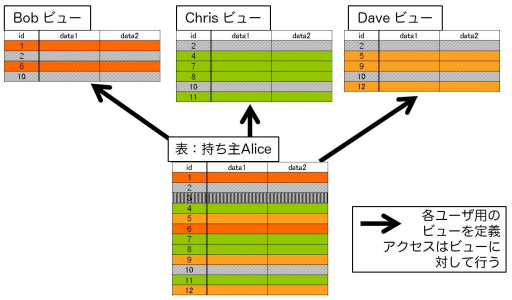

図 6 ビューを利用したアクセス制御

で、利用時に Facebook などの SNS のアカウントを用いて認証を行い、異なる LAN に存在するユーザの計算機同士を簡単に通信可能にする仕組みである。SNS のユーザの計算機は、自分の計算機を接続しているルータの向こう側に存在するように見える。ソーシャルルータは DNS サーバの機能を持ち、各 SNS ユーザの計算機には特別のドメイン名と IP アドレスを割り当てる。例えば、SNS ユーザ Alice のドメイン名は alice.socialrouter のように割り当てる。ソーシャルルータでは、IP アドレスやドメイン名から SNS のユーザ名を知ることができる。加えてソーシャルルータには強力なアクセス制御機能が存在する。TCP/IP のポート単位で通信を制御できるため、SNS のユーザは許されたサービスにのみ接続することができる。例えば、HTTP のみを許可したり、MySQL のみを許可したりできる。

ソーシャルルータは本研究室で開発を行なっている。アプリケーションは、Network File System 等の LAN 用アプリケーション、Internet Relay Chat のようなインターネット用のアプリケーション、本研究で述べるソーシャル SQL データベース及びネットワークニュース Friend News System がある [4]。Friend News System では、ネットワークニュースの記事をソーシャルルータが提供する通信路で配信する。

2.4 データベースのアクセス制御

ソーシャル SQL データベースは SNS 上の交友関係に基づいた通信路を利用するが、自分自身の計算機に持つデータベースに接続してくる各ユーザごとにアクセス制御が必要である。ソーシャルルータは友人関係にあるユーザ同士を通信可能にする。しかし、ユーザにより社会的なつながり方や信頼の度合いは異なる。それらに応じて各ユーザが自身の計算機に持つデータベースへのアクセス制御を行うことが出来れば有用である。例えば、社会的なつながりは家族、友人、趣味、ビジネスなど多様であり、信頼の度合いもまた、家族同然に親しい、職場や学校で仲が良い、顔を知っている程度など多様である。ソーシャルルータではポート番号を用いて SQL サーバへのアクセスの可否を判断することができるが、これらのつながりや度合いを同等に扱うことになり、各ユーザが自身の計算機に持つデータベースにアクセス制御を行うだけでは不十分である。よってソーシャル SQL データベースではソーシャルルータではできない細かいアプリケーションレベルのアクセス制御を各ユーザが自身の計算機に持つデータベースに対して提供する。

本研究では、各ユーザが自身の計算機に持つデータベースの表に対して、アクセスの権限に応じたビューを生成する。図 6 にその概略を示す。図 6 では Alice のデータベースに Bob がアクセスした場合、元の表から Bob の権限でアクセスできるビューを生成している。

ソーシャル SQL データベースのモジュール構成を図 7 に示す。全体としてシステムは SQL クライアント、SQL サーバ、プロキシ、トランスポート層およびビュー設定ユーティリティから構成される。ソーシャル SQL データベースの中核はプロキシである。これはクライアントから要求を受け取り、クライアントを判定しそれに応じたビューに接続先を書き換えて SQL サーバに中継する。ビュー設定ユーティリティは、SNS のユーザごとに、SQL サーバに対してビューの定義を与え、プロキシに対してはビューとユーザの対応付けを与える。トランスポート層としてソーシャルルータを用いることで、そのユーザ認証機能を利用する。クライアントは、2.2 節で述べたように JavaScript API または TCP/IP を用いて通信する。SQL サーバは、直接外部ネットワークからアクセスすることを禁止し、プロキシを通過しないとアクセスできないようにする。これにより、不正なアクセスを遮断しデータを守ることができる。ソーシャル SQL データベースでは、自分に計算機のデータベースにアクセスする際は、プロキシを経由することはない。

3. MySQL によるソーシャル SQL データベースの実現

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。各ユーザが自身の計算機に持つデータベースには以下のものを用いる。

- MySQL 5.1.63
- MySQL Proxy 0.8.3

MySQL Proxy[5] は、MySQL のクライアントとサーバ

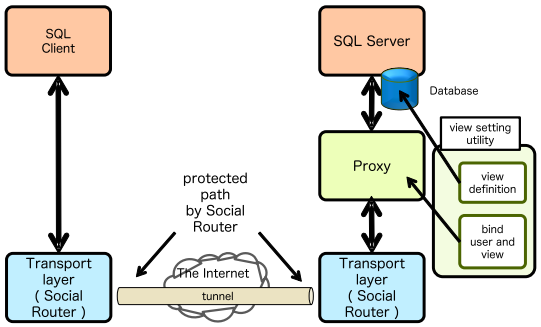

図 7 ソーシャル SQL データベースの構成

の通信を中継するソフトウェアである。MySQL Proxy は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。

3.1 MySQL DB の作成

2 章では、ユーザごとにビューを生成し、それを用いてアクセス制御を実現すると述べた。しかしながら、MySQL と MySQL Proxy を用いて単純にこの方法を実現しようとすると、MySQL Proxy において SQL 文の構文解析を行い SQL 文を書き換えなければならない。この書き換えを実現するためには複雑な SQL 文を解析する必要があり、Lua で拡張すると実装上の問題がある。

この問題を解決するために本研究では、MySQL が持っている複数の表をまとめて扱う仕組みである database（以後、**MySQL DB** と呼ぶ）を用いる。MySQL のクライアントは拡張された構文 CREATE DATABASE を用いて MySQL DB を作成することができる。wordpress という名前の MySQL DB を作成する例を以下に示す。

```
CREATE DATABASE wordpress
```

MySQL のクライアントは拡張された構文 USE を用いて MySQL DB を切り替えることができる。wordpress という名前の MySQL DB に切り替える例を以下に示す。

```
USE wordpress
```

なお MySQL DB と同様の機能は他の関係データベースである Oracle や PostgreSQL 等にも実装されている。したがって本手法は汎用性があり、それらの関係データベースでも利用可能である。

本研究では、クライアントが接続してきた時に SNS のアカウントに応じて MySQL DB を切り替えることによりアクセス制御を実現する。そのためにはまず、サーバ側でクライアントごとに MySQL DB を作成する。MySQL DB とクライアントの対応は例えば表 1 のようになる。表 1 において Social Domain name とはサーバのデータベースに接続してくるクライアント計算機のドメイン名のことであり、SNS のアカウントと結びついている。MySQL account とは、SQL サーバに接続するために MySQL Proxy によって書き換えた後のログイン ID である。src_MySQL DB とは、クライアントが指定してくるアプリケーションのデータベースである。MySQL account と src_MySQL DB の組みにより実際にクライアントに接続させるデータベース、dst_MySQL DB を決定する。そしてクライアントにアクセスさせる表に対してアクセスの権限に応じたビューをクライアントごとの MySQL DB 配下に生成する。これらのビューの名前はオリジナルの MySQL DB の表の名前と同一にする。MySQL DB と配下のビュー設定はビューの設定ユーティリティで行う。ビュー設定ユーティリティで SQL サーバの管理者は、アクセス制御のために各 MySQL DB 配下にビューを定義する。

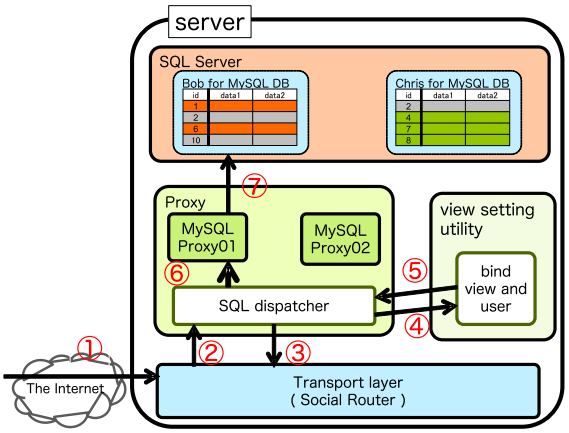

図 8 プロキシのよる接続元問い合わせ

Alice が Bob のためのビューを作成するための SQL 文の例を以下に示す。

```
USE Bob_wordpress
CREATE VIEW wp_posts AS
SELECT * FROM wordpress.wp_posts
WHERE NOT category = 'fishing';
```

この例では、Alice はブログデータを保存した表 wp_posts のうち fishing 関連のものだけは Bob に見られたくない。そのため元の表 wp_posts から fishing 関連のデータだけ取り除いたビュー wp_posts ビューを作成している。以上のような実装をそれぞれのユーザが自分の計算機のデータベースに施し他のユーザから利用可能にする。

3.2 MySQL DB の切り替え

各ユーザは、自分の MySQL サーバには SNS のユーザごとに権限を制限したアカウントを作る。そのアカウントでは、3.1 項で作成した MySQL DB として元のデータベースのビューをアクセスできるようにする。図 8 に MySQL DB を切り替えるための手順を示す。MySQL のプロキシの手前に、SNS のユーザからの要求を受け付けるディスパッチャを置く。ディスパッチャは SNS のユーザごとに、MySQL のプロキシのインスタンスを実行する。

本研究では、図 8 のように遠隔クライアントの MySQL DB を切り替える。

(1) SQL クライアントは、TCP/IP で SQL ディスパッ

表 1　MySQL DB とクライアントの対応付け

Social Domain name	MySQL account	src MySQL DB	dst MySQL DB
pc01.Bob.socialrouter	Bob	wordpress	Bob_wordpress
tablet.Chris.socialrouter	Chris	wordpress	Chris_wordpress
lab.Dave.socialrouter	Dave	wordpress	Dave_wordoress

チャに接続する。この際、SQL クライアントは固定のユーザ名と固定のパスワードで接続を要求する。

(2) SQL ディスパッチャは、TCP/IP で接続を受け付ける。SQL ディスパッチャはクライアントの IP アドレスを得る（図 8 の 2)。

(3) SQL ディスパッチャは IP アドレスをソーシャル・ルータに問い合わせて、ドメイン名を得る。SQL ディスパッチャはドメイン名から、SNS のアカウント名を得る（図 8 の 3)。

(4) SQL ディスパッチャは、ビュー設定ユーティリティから得られたクライアントの SNS アカウントに対応した MySQL Proxy の接続先を決定する。(図 8 の 4 と 5)。

(5) SQL ディスパッチャは、SNS のアカウント名を用いて対応する MySQL Proxy のインスタンスに TCP/IP で接続する。以後、SQL ディスパッチャは、SQL クライアントと SQL プロキシの間を TCP/IP のレベルで中継する。この際、通信内容は解釈しない（図 8 の 6)。

(6) MySQL Proxy は、SQL ディスパッチャ経由で SQL クライアントから受け取った固定のユーザ名と固定のパスワードを捨てる。ビュー設定ユーティリティから得られた MySQL DB のアカウント名とパスワードに書き換えて、MySQL サーバに接続する（ 図 8 の 7)。

(7) SQL クライアントは、SQL 文で要求を SQL ディスパッチャに送る。SQL ディスパッチャは、要求をそのまま MySQL プロキシに送る。MySQL プロキシもまた要求をそのまま MySQL サーバに送る。MySQL サーバは、SQL 文を実行する。結果は逆順に SQL クライアントに返される。

固定のユーザ名とパスワードを利用する利点は 2 つある。1 つ目は、クライアントはログインする際にユーザ名とパスワードを入力する必要がなくなる。このことによりサーバ側もユーザ名とパスワードをクライアント側の SNS 利用者に配布する手間もなくなる。SNS アカウントと MySQL ユーザが結びついているのでクライアントはソーシャルルータの利用開始時に認証が済んでいる。サーバは SNS アカウントが信用できる限りパスワード無しで適切な MySQL DB に中継しても良い。2 つ目は、サーバ側もログインしてくるユーザのログイン名は SNS アカウントと結び付けられれば自由に決定できることである。

本研究では、Lua のプログラムを用いて MySQL Proxy を拡張している。図 9 に MySQL Proxy 拡張のための Lua 言語のプログラムを示す。

このプログラムは 1 つ目の機能としてプロキシは、SNS のアカウントに応じて MySQL データベースに接続するための認証情報を書き換える。具体的に書き換えるものはログイン ID とパスワードである。図 9 の read_auth は、クライアントが MySQL サーバにログインする際に MySQL Proxy において呼ばれる Lua の関数である。proto.queries.append 関数は、クライアントからサーバに送信される SQL クエリに、MySQL Proxy が新たな SQL クエリを追加する関数である。図 9 では、proto.to_response_packet 関数を使って SQL クエリを作成している。proto.to_response_packet 関数の引数は、username、response、charset、database、max_packet_size 及び server_capabilities である。この username は、MySQL Proxy で書き換えたログインユーザ名である。username は、ビューとユーザの名前を参照して書き換える。get_mysql_user で取得する。response とは、ログインユーザの暗号化されたパスワードである。クライアント固定の値は捨てられるので、すべてのユーザにおいて共通である。以下のパラメータは、クライアントから送信されたものを変更せずに用いる。charset とは、文字のエンコード方式である。database とは、接続先 MySQL DB である。max_packet_size とは、送受信データの最大のパケットサイズを表す。server_capabilities とは、サーバにおける各種の権限を表す。

2 つ目の機能としてプロキシは、SQL の拡張構文 USE を検知すると、強制的に接続先をクライアントが本来接続すべき適切なものに書き換える。USE 構文は接続先 MySQL DB を切り替えるものなので、切り替え先は SNS アカウントに応じたクライアントが接続すべき SQL サーバの MySQL DB となる。図 9 の read_query とは、認証後のクライアントが MySQL サーバに SQL クエリを送信する際に MySQL Proxy において呼ばれる Lua の関数である。read_query は、引数に packet をとる。packet は、クライアントからサーバへ送信された SQL のデータグラムである。packet に proxy.COM_INIT_DB が含まれていた場合、ビューとユーザの対応を参照して、MySQL Proxy は接続先 MySQL DB を書き換える。proxy.COM_INIT_DB が MySQL Proxy においてデータグラム中の USE 構文を表す。

このような方法を取ることでアクセス制御のために SQL 文の構文を解析し表をビューに書き換えるという手段を用

いる必要がなくなる。結果を取得する際、クライアントは途中で要求が書き換えられたことを知ることはない。またクライアントはどの MySQL DB にアクセスするか気にすることもなくなる。SQL サーバでは、SNS アカウントに応じてクライアントが操作できる MySQL DB を限定するため、アクセス制御を実現できる。

3.3 ビューを利用したことによる制限事項

ソーシャル SQL データベースではクライアントのアクセス制御のため、複数の表に対し結合演算、選択演算及び射影演算を行い作成したビューに接続させている。このため MySQL を使った実装では、操作に制限が生じる。例えば、複数の表に対して結合演算を行い作成したビューに対しては、ビューの要素を作成・更新・削除することができないという制限がある。一方、他の表との結合演算を行わず、単一の表に対してのみ選択演算を行い作成したビューに接続させた場合はビューの要素を作成・更新・削除を行うことができる。

3.4 想定されるセキュリティ上の問題とその対処方法

クライアントが USE 構文を使って勝手に MySQL DB を切り替えてしまう攻撃が考えられる。クライアントは接続するときサーバで使用する MySQL DB を指定する。例えば図 7 で、Alice がサーバ側で Bob がクライアント側であるとき Bob が実行するクライアントは以下のように他人の MySQL DB を指定することが考えられる。

```
USE Chris_wordpress
```

この問題の解決策として、MySQL Proxy で USE 構文のリクエストを検出すると強制的に接続先 MySQL DB を書き換える。

また MySQL では、MySQL DB_name.table_name と記述することで USE で指定した MySQL DB 以外にアクセスできる。このため、他人の MySQL DB にアクセスされてしまうという問題がある。この問題の解決策として、SNS アカウントと対応させて各ユーザごとに権限を絞り込むことで対処している。ここでいう権限の絞り込みとは各ユーザが SQL サーバで使うアカウントで使用できる MySQL DB を限定することである。各ユーザごとに SQL サーバにアクセス制御を行ったアカウントを接続することで許可されていない他人の MySQL DB にアクセスされることを防ぐ。

クライアントごとに MySQL DB にアクセスするためのアカウントを作成し、そのアカウントのユーザ名でローカルデータベースにアクセスさせる。各アカウントでは予め指定した MySQL DB 以外のアクセスは禁止しているので、MySQL DB_name.table_name のように他人の MySQL DB にリクエストが来た場合はエラーを返す。Alice が Bob に対してビュー Bob_wordpress に権限を作成する例を示す。

```
1  function read_auth(auth)
2    ...
3    proxy.queries:append(1,
4      proto.to_response_packet({
5        username = get_mysql_user(),
6        response = fixed,
7        charset  = original.charset,
8        database = original.database,
9        max_packet_size = original.maxpacketsize,
10       server_capabilities = original.servercap
11     })
12   )
13   ...
14 end
15 function read_query( packet )
16    if string.byte(packet) == proxy.COM_INIT_DB
          then
17     view = view_search()
18     proxy.queries:append(2, string.char(proxy.
           COM_INIT_DB) .. view, {resultset_is_needed
            = true})
19   ...
20 end
```

図 9 Lua による認証情報と MySQL DB の書き換え

```
GRANT ALL ON Bob_wordpress.* TO Bob
IDENTIFIED BY 'passwd'
```

4. Web ブラウザ JavaScript に対する API の実装

ソーシャル SQL データベースを 2.2.3 項で述べた Web ブラウザ上で動作するアプリケーションから利用可能にする。そのために本研究では、Web ブラウザで広く利用されている JavaScript で記述されたアプリケーションに対して SQL の API を提供する。

最近の Web ブラウザで動作する JavaScript からは HTML5 が利用できる。HTML5 は WHATWG および W3C が標準化を進めている次世代の HTML 規格である。HTML5 では Web SQL Database という機能が検討された [6]。これは各ユーザごとに自身の計算機にデータベースを持ち、そこにブラウザ上で動作する JavaScript のアプリケーションがデータを保存できるというものである。データベースを操作するための JavaScript API も定義されている。

しかし HTML5 で検討された Web SQL Database では、Web ブラウザが動作している計算機の SQLite による関係データベースを操作するだけで TCP/IP で任意のサーバにアクセスする機能が無い。Firefox 等の Web ブラウザの Addon の API では、TCP/IP のソケットを使うことができるが SQL データベースに接続するためのインターフェースがない。例えば、Java には、様々なデータベースと接続を橋渡しする JDBC（Java DataBase Connectivity）と

いう API があり Java 言語からはこれを利用しデータベースを操作する。他にも RubyDBI 及び PHP Data Objects といったものがあるが、ブラウザで利用可能になっている JavaScript には、そのようなデータベースへのインタフェースが存在しない。

この問題を解決するために本研究では JavaScript から Java 言語用のインターフェースを JavaScript から利用可能にする。JavaScript からデータベースを操作する一連の流れを図 10 に示す。まず JavaScript から Java Applet を利用する。Java Applet からは、JDBC を用いてデータベースに接続し操作を行う。結果はデータベース、JDBC、JavaScript と逆に返ってくるため JavaScript で処理する。

この JavaScript API では、複数のユーザのデータベースを同時に利用するアプリケーションを実現可能にする。これにより Web サーバを用いずに、Web ブラウザをユーザインタフェースとして SQL データベースを利用するソーシャル・アプリケーションを作成することができる。

表 2 に、本研究で作成した JavaScript API を示す。connectDatabase は、Web SQL Database の openDatabase に相当する。openDatabase は、動作している Web ブラウザ上の SQLite データベースのみ操作できる。対して本研究の API は、Web ブラウザが動作している計算機外のデータベースを操作できるという違いがある。また SQLite 以外のデータベースにも接続することができる。executeSql は、Web SQL Database の同名の関数 executeSql に相当する。close は、データベースとの接続を明示的に終了するために使用する。

図 11 に、SQL データベースを利用する JavaScript のコードの例を示す。まず<applet>タグを用いて、Java Applet を読み込む。connectDatabase() を用いて、SNS のユーザのデータベースに接続する。引数は、SNS のユーザの端末のドメイン名、ポート番号及び使用するデータベース名である。本研究では、SNS アカウントを用いて認証を行うため、ログインユーザ名とパスワードを入力する必要はない。次にこのコードは、executeSql() を用いて、SQL 文の要求を送信し、結果を取得する。最後に、close() でデータベースとの接続を終了する。この API は、Web SQL Database と類似の操作を提供する。

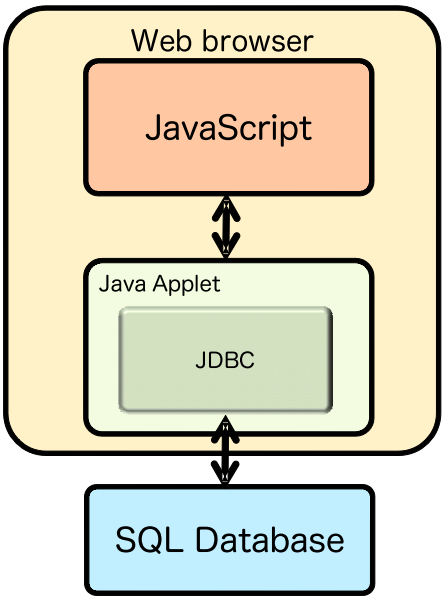

図 10 Web ブラウザ用の JavaScript API の構造

```
1  <html>
2  <head>
3    <title>Java Applet Connection DB</title>
4  </head>
5  <body>
6  <applet id="mysql-connection" code="MySQL.class"
       codebase="./" archive="mysql-connector-java
       -5.1.26-bin.jar"></applet>
7  <SCRIPT src="connectdatabase.js"></SCRIPT>
8  <SCRIPT type="text/javascript">
9  var db = connectDatabase("Alice.socialrouter",
       "3306","calender");
10 var result = db.executeSql("SELECT * FROM
       my_schedule WHERE datetime BETWEEN
       '2013-10-01' AND '2013-10-31'");
11 db.close();
12 </SCRIPT>
13 </body>
14 </html>
```

図 11 JavaScript のソースコード

5. Web アプリケーション

本研究ではソーシャル SQL データベースと Web サーバを用いるアプリケーションを連携させることでその有用性を評価する。アプリケーションの例を 2 つ示す。いずれも構成は図 4 と同様でアプリケーション部分のみ異なる。

5.1 WordPress

1 つ目の Web アプリケーションはブログアプリケーションである。ブログアプリケーションは Web 上のアプリケーションであり、記事の投稿やそれに対するコメントを行うなど協調作業の要素を持つ。また記事に対して公開、認証付き公開、非公開といった設定を施すなどアクセス制御の面でも、ソーシャル SQL データベースを利用する例として適切である。

ブログアプリケーションとしては、WordPress[7] を用いる。WordPress はオープンソースのブログアプリケーションであり、データベースとして SQL データベースを使用する。主な特徴として、PHP を用いた動的な Web ページ生成、投稿記事へ複数のカテゴリを設定可能、プラグインによる拡張機能などが挙げられる。本研究では、WordPress をソーシャル SQL データベース使うようにする。接続データベースを MySQL Proxy で変更するだけなのでアプリ

表 2　JavaScript API の説明

API	説明
connectDatabase(host,port,database)	相手のデータベースと接続するための関数である。引数は、host、port 及び database(MySQL の場合、MySQL DB) である。結果として、指定されたデータベースを操作するためのオブジェクトを返す。
executeSql(sqlStr)	SQL 文を実行し、結果を返す関数である。引数 sqlStr は、SQL 文である。結果の型は、JSON の配列形式である。
close()	相手のデータベースとの接続を終了する関数である。引数は取らない。

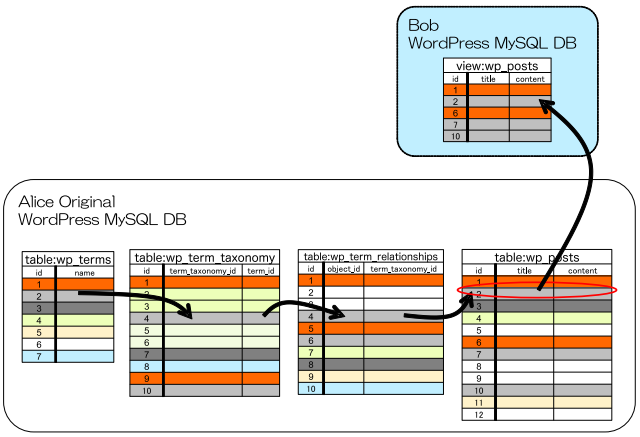

図 12　WordPress におけるビューの作成

ケーション自体に手を加えることはない。

現在までにアプリケーションの接続ユーザ名に応じたブログ投稿記事の閲覧アクセス制御ができている。記事の分別については WordPress の記事をカテゴリに分ける機能を利用し、そのカテゴリに一致したかしないかを条件にして予めビューを作成している。図 12 に作成するための例を示す。このビューを作成するに当たり、まずオリジナルの表 4 つ、wp_terms,wp_term_taxonomy,wp_term_relationships 及び wp_posts で JOIN 演算を行う。これらの表を JOIN 演算する理由は、カテゴリで選択したもののみの投稿ビューを構成する時に、カテゴリ情報と投稿情報を結びつけるために最低限必要だからである。次に指定カテゴリにより選択演算を行う。最後にオリジナルの表と同じ構造になるように射影演算を行う。

MySQL のユーザ名を切り替えるだけでそれに応じた MySQL DB に接続させることが可能である。これにより接続ユーザが変わるだけでユーザごとに違う投稿記事を閲覧させることができる。図 13 に Alice が、自計算機上の SQL クライアントから Chris の SQL サーバにアクセスした場合を示し、図 14 に Bob がアクセスした場合を示す。Alice は、'daily' カテゴリに含まれる記事を閲覧することができる。Bob は、'game' カテゴリに含まれる記事を閲覧することができる。図 13、図 14 において、「ソーシャルルータ」とタイトルが付いている記事は、'daily','game' 両

図 13　WordPress：Alice が閲覧した場合

方のカテゴリに属しているため、Alice、Bob 両者が閲覧することができる。このように、本研究の機能を用いれば、WordPress が本来使用する機能（公開、認証付き公開、非公開）よりはるかに細かくアクセス制御が実現できる。なお接続ユーザは投稿記事についてコメントを書くことができる。3.3 節で述べた制約により、投稿記事を作成・更新・削除することはできない。

5.2　写真共有アプリケーション

2 つ目のアプリケーションは、写真のアルバムアプリケーションである。アルバムアプリケーションは簡易的なものを Web アプリケーションフレームワークの Ruby on Rails を用いて作成した。各写真の閲覧制御については ACL（Access Control List）を利用し、それを条件にして予めビューを作成している。ACL によるビューの作成例を以下に示す。

```
CREATE VIEW ‘photos’ AS (
SELECT ’photos’.’title’,
’photos’,’msg’, ’photos’.’photo’
```

図 14 WordPress：Bob が閲覧した場合

図 15 写真アルバム：Alice が閲覧した場合

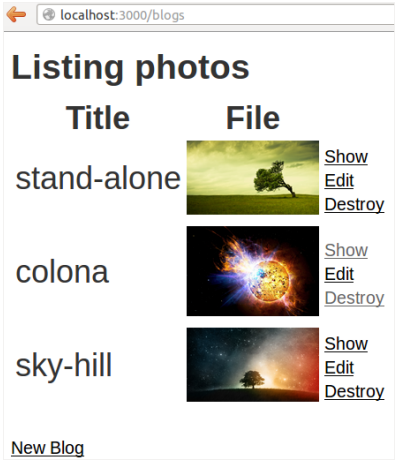

図 16 写真アルバム：Bob が閲覧した場合

```
FROM ‘photo‘.‘photos‘ JOIN ’photo’.’ACL’
WHERE ’photo’.’photos’.’id’ = ’photo’.’ACL’.’acl_id’
AND ’photo’.’ACL’.’user’ = ’Bob’);
```

この例では、Bob のためのビューを作成したいため、ACL の値に Bob が含まれたレコードのみでビューを作成する。ACL は、専用の表を用意し参照する。その表と JOIN 演算、選択演算を行うことで特定ユーザ（ここでは Bob）のみが閲覧できるビューを作成することができる。ここで photo とはオリジナルの MySQL DB である。この MySQL DB には、表 photos と表 ACL が存在する。ここで作成しているビュー photos は、オリジナルの表と同一の名前にする。

このように MySQL のユーザ名を切り替えるだけでそれに応じた MySQL DB に接続させることが可能である。図 15 に Alice が、自らの計算機上の SQL クライアントから Chris の SQL サーバにアクセスした場合を示し、図 16 に Bob がアクセスした場合を示す。図 15、図 16 において写真「stand-alone」は、ACL で Alice、Bob 両者に許可があるため、両者閲覧することができる。このように接続ユーザが変わるだけでユーザごとに違う写真を閲覧させることができる。接続ユーザは写真を投稿（作成）・更新・削除することができる。接続ユーザが投稿等をした場合、ACL に強制的に接続ユーザが追加されるため、データベースの所有者が更新・削除できなくなることはない。

5.3 実験環境

本研究でソーシャル SQL データベースと各アプリケーションを動作させるために構築した実験環境を図 17 に示す。実験環境はひとつの実機計算機上に仮想計算機モニタを導入し、その上に仮想ネットワークと仮想計算機を配置することによって実現している。まず仮想計算機モニタ上に複数の独立した LAN を構築する。それぞれの LAN にひとつずつソーシャルルータを配置し、The Internet のゲートウェイとする。それぞれの LAN に実験用の仮想計算機を配置する。これらの仮想計算機はそのままでは異なるネットワークの仮想計算機と通信することはできない。これらの仮想計算機がブラウザ上で Facebook の認証を通過した時、初めて異なる LAN に属していてもお互いの SNS アカウントが友人関係ならば仮想計算機同士で通信することが可能になる。ソーシャルルータ及び各仮想計算機には外部から制御するために SSH サーバが起動しており、実験を行うときは SSH を用いて各計算機を管理する。

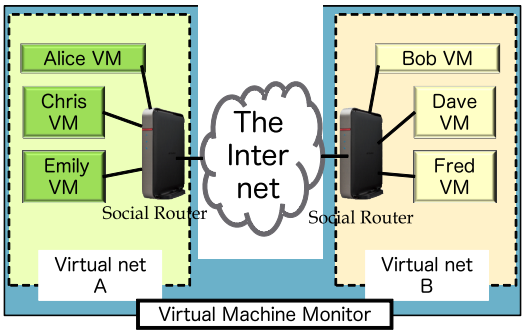

図 17 ソーシャルルータを用いた実験ネットワーク

6. 関連研究

Oceanstore[8] は分散型ストレージのインフラストラクチャのひとつである。Oceanstore はすべてのオブジェクトが暗号化されている。暗号鍵をユーザが共有することでオブジェクトを共有することができる。Oceanstore は Access Control List (ACL) を使ってアクセス制御を行う。サーバ上のデータをすべて暗号化する点では CryptDB[9] も同様である。データベースの問い合わせ言語としては、SQL を基に設計された特別な SQL を通信に用いる。本研究では、各ユーザがデータを自分の計算機に保存し、信頼できる友人のみアクセスを許可するのでデータを暗号化する必要がないという点で異なる。

HomeViews[10] はアクセス制御の仕組みとしてケーパビリティを用いている。HomeViews ではビューに対してケーパビリティを定義する。ケーパビリティを譲渡していくことで別のユーザにアクセス権を譲渡することができる。本研究では、アクセス制御にケーパビリティではなく、ACL の一種である「ビューとユーザの対応」を用いている点で異なる。

7. おわりに

本研究では、中央サーバを用いることで生じるプライバシ侵害の問題を解決する方法としてソーシャル SQL データベースを提案した。このソーシャル SQL データベースは SNS のユーザ間の通信にソーシャルルータを利用し信頼のおけるユーザ間で相互にデータベースを利用可能にする。ソーシャルルータによって SNS を利用しユーザを判別して、それぞれに応じたビューを応答することでアクセス制御を実現している。Web ブラウザで動作する JavaScript のアプリケーションに対しては、Java Applet を用いてデータベースを操作する API を提供する。MySQL を用いてソーシャル SQL データベースを利用するアプリケーションとしてブログ WordPress と写真アルバムアプリケーションを示した。接続するクライアントに応じて閲覧できるデータが異なることを示した。

今後の課題は、2 つある。1 つはビュー設定ユーティリティが未完成なため完成させることである。もう 1 つは、ブラウザ用の API でコールバック関数を利用可能にすることである。

参考文献

[1] Datta, A., Buchegger, S., Vu, L.-H., Strufe, T. and Rzadca, K.: Decentralized Online Social Networks, Springer Science+Business Media, pp. 349–378 (2010).
[2] 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).
[3] 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*, pp. 278–286 (2011).
[4] 肖焜瑤, 新城靖, 櫻井孝一, 佐藤聡, 須藤侑一, 中井央: ソーシャルルータを用いたネットニュースシステムの実現, 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] MySQL 5.6 Reference Manual: MySQL Proxy (2012). `http://dev.mysql.com/doc/refman/`.
[6] Hickson, I (ed.): Web SQL Database (2010). `http://www.w3.org/TR/webdatabase/`.
[7] WordPress Reference Manual: WordPress (2013). `http://codex.wordpress.org/`.
[8] Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C. and Zhao, B.: OceanStore: an architecture for global-scale persistent storage, *ACM SIGPLAN Notices*, Vol. 35, No. 11, pp. 190–201 (2000).
[9] Popa, R. A., Redfield, C. M. S., Zeldovich, N. and Balakrishnan, H.: CryptDB: protecting confidentiality with encrypted query processing, *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pp. 85–100 (2011).
[10] Roxana, G., Magdalena, B., Gribble, S. D. and Levy, H. M.: Homeviews: peer-to-peer middleware for personal data sharing applications, *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, SIGMOD '07, pp. 235–246 (2007).