組込型データベースとキーバリュー型データベースの画像連携に関する初歩的研究
―どこでも画像アップロードの試作―

島根大学　総合理工学部　数理・情報システム学科
計算機科学講座　田中研究室
Ｓ０８３１０５　吉留　明樹
平成２４年　２月２４日

目次

１.序論

２.組込型データベースとは

2.1　スマートフォンと組込型データベース
2.2　SQLite

３.キーバリュー型データベースとは

3.1　クラウドコンピューティングとキーバリュー型データベース
3.1.1 クラウドコンピューティング
3.1.2 キーバリュー型データベース
3.2　Google　App　Engine
3.2.1Google　App　Engine とは
3.2.2Google　App　Engine のメリット
3.3　Bigtable と Datastore

４.どこでも画像アップロード

4.1　システムの設計
4.1.1　システムの仕様
4.1.2　製品仕様
4.2　システムの内容
4.2.1　概要
4.2.2　クライアント側
4.2.3　サーバ側
4.2.4　実行画面

５.結論

謝辞

引用文献

１.序論

データベースは今まで、クライアント/サーバ型のリレーショナルデータベース管理システム(以下、RDBMS)が主流であった。しかし、Android 等を OS とするスマートフォンや Google App Engine(以下、GAE)等といったクラウドコンピューティングサービスが急速に普及する現在では、「組込型データベース」と「キーバリュー型データベース」と呼ばれる２つのデータベースが RDBMS に代わる新しいデータベースになりつつある[1]。

本研究では、代表的なマルチメディアデータである画像を上記で挙げた２つのデータベースの「組込型データベース」と「キーバリュー型データベース」を使用し画像連携を行うシステムである「どこでも画像アップロード」を試作した。

２.組込型データベースとは

2.1 スマートフォンと組込型データベース

現在、スマートフォンが急速に普及している。スマートフォンとは、携帯電話にパソコンや PDA の機能が備わった多機能携帯電話である。汎用の OS を搭載しており、アプリケーションソフトをダウンロードすることができる。このスマートフォンで使われているデータベースが組込型データベースである[1]。

組込型データベースとは、組込みシステムに特有の制約・設計思想に基づいたデータベース管理システム(以下、DBMS)である。厳しいリソース制約上での稼働、ゼロアドミニストレーション、高い信頼性、リアルタイム性などが特徴として挙げられる。組込型データベースは、トランザクション処理に対応した RDBMS であるが、クライアント/サーバ型のデータベースではなく、クライアントとサーバが一体となったサーバーレスのデータベースである。

組込系のものは、組込型データバースが開発されるまでデータはファイルによって管理されていた。しかし、携帯電話など組込系の技術が進化していくにつれてデータの量や種類などが増えていき、データを効率的に利用することかつ処理するために組込系でも組込型データベースと呼ばれるデータベースが開発された。

そして、スマートフォンの代表的な OS である Android や iOS で使われている組み込み型データベースが「SQLite」と呼ばれる組込型データベースである

2.2　SQLite

SQLite は、D.Richard Hipp が開発した RDBMS の組込型データベースの一つであるが、従来の RDBMS である MySQL や PostgreSQL のような多くのユーザーとデータを共有できるクライアント/サーバ型ではなく、クライアントとサーバが一体化しているサーバーレスであり、アプリケーションに組み込んで利用する軽量のデータベースである。また、「保存されたデータは一つのファイルにまとめられるファイルベースである」、「型の扱いが柔軟である」、「トランザクション処理に対応している」などの特徴がある[1]。

SQLite のメリットは、次の５つが挙げられる。バックアップが容易である。レンタルサーバの MySQL と比較すると動作が速い。他のアプリケーションとデータを共有できる。データ型の概念が希薄であり、長さの指定が必要ない。設定や納入が容易である。

まず、「バックアップが容易である」だが、特徴で述べたようにデータが一つのファイルにまとめられているファイルベースであるため、サーバ上でコピーを行えば容易にバックアップができる。次に「レンタルサーバの MySQL と比較すると動作が速い」であるが、レンタルサーバを使っている場合、多くのユーザーも一緒に使用していることがあるためサーバに負担がかかり動作が遅くなるが、SQLite なら自分のサーバ上に置くため動作が遅くならない。次に「データ型の概念が希薄であり、長さの指定が必要ない」についてだが、特徴で述べたように型の扱いが柔軟であるため型指定のないカラムを作れたり誤った方を挿入しようとしたとき適当に型を解釈したり、また、従来のデータベースはテキストを定義する際長さを指定しなくてもいいのである。次に「設定や納入が容易である」だが、パスワードを設定する必要がなくすぐ利用でき、プログラムをコピーするだけで納入できる。

このように多くのメリットがあるがデメリットもある。SQLite がファイルベースのデータベースであるためユーザーの管理の概念がないので SQLite にはセキュリティ機能がないためパスワードの設定ができない。SQLite ではデータがローカルに保存されるため複数のサーバからアクセスできない。バージョンの互換性がない。

Android で SQLite が採用された理由は、３つある。

・軽量であること
・組込システム用に最適化されている
・データを共有できること

スマートフォンは、パソコンをベースにしているため全体的にシステムが重いため他の組込型データベースと比較して軽量なデータベースであるため SQLite を使用している。SQLite は、スマートフォン等といった組込システム用のデータベースとして使用するためにかいはつされたため最適化されているのである。

また、Android は Android マーケットと呼ばれる Android 用アプリケーション配信サービスからアプリケーションをダウンロードして自分専用のスマートフォンにしていく。そこで、アプリケーション同士でデータを共有する必要があるため、データを共有でき

る SQLite が使用されている。データとしての例は、氏名/電話番号/住所などのアドレスデータや画像データ、画面メモなどが挙げられる。これらの理由で SQLite は採用されたのである。

3.キーバリュー型データベースとは

3.1 クラウドコンピューティングとキーバリュー型データベース

3.1.1 クラウドコンピューティング

クラウドコンピューティングとは、あらかじめ用意された巨大なサーバをインターネット等のネットワークを通じて、サービスの形でデータやソフトウェアを必要に応じて利用できるシステム形態である。そして、このクラウドコンピューティングを利用したサービスをクラウドコンピューティングサービスといい、主に、次の３つに分類される[2]。

- IaaS(Infrastructure as a Service)

  データセンターのようなサーバ群の一部を、Web サーバなどのインフラとして提供するサービスである。

  仮想化技術を使ってスーパーコンピュータのような処理能力を利用できるようにしたり、１台のコンピュータを複数のコンピュータとして仮想的に分割して複数のユーザーが共有できるようにしたりと仮想化されたサーバ上でハードウェアや OS が提供される。ユーザーは、新しくハードウェアを増設する必要が無いが、ハードウェアがどんなもので OS やソフトウェアが何を使っているのか知らなければならない。

  代表的なものとして、Amazon 社が提供している Amazon　EC2 である。

- SaaS(Software as a Service)

  ネットワークを通じてアプリケーションを利用できるようにしたサービスである。システムの構築や運用について気にする必要がなく、また、アプリケーションは自分でカスタマイズすることができるので必要な機能だけ利用できる。

  代表的なものとして、salesforce.com 社が提供している顧客管理システムの salesfoce.com である。

- PaaS(Platform as a Service)

  ネットワークを通じて、アプリケーションを開発・利用するために大量のコンピュータ群を１つのプラットフォームとして提供するサービスである。

  PaaS では、ハードウェアや OS、Web サーバ、データベースサーバが何を使用しているのかユーザーは知る必要がなく、それらを運用するための知識も必要ない。ユーザーは、企業から提供される開発用ライブラリ（ＳＤＫ）を利用してアプリケーションを開発するだけである。サイトへのアクセスが急激に増加してサーバへの負荷が増大してもユーザーは気にする必要がなく、自動的にサーバの負荷に応じて処理を行う。

  代表的なものとして、Google 社が提供している Google　App　Engine がある。

これら、クラウドコンピューティングの中のデータを管理し、実現するために必要不可欠なのがキーバリュー型データベースである[2]。

3.1.2 キーバリュー型データベース

キーバリュー型データベースとは、データの保存・管理手法の一つで、任意の保存したいデータ(value)に対し、対応するものの名前(key)を設定し、これらをペアで保存する方式をとったデータベースのことである。

キーバリュー型データベースでは、保存したいデータに他のデータと識別するための文字列や数値を割り当て、データとセットで記憶装置などに保管する。データの読み出し時には、予め設定したキーを指定すると、対応したデータを取り出すことができる。保存できるデータの種類はソフトウェアによって異なる。また、様々なプログラミング言語で連想配列、ハッシュなどと呼ばれるデータ型の仕組みをデータ管理システムに応用したものである。

キーバリュー型データベースのメリットは、以下の点が挙げられる。

・高いスケーラビリティ

・高可用性

キーバリュー型データベースは、キーとバリューというシンプルな組み合わせでデータを保存・管理するためスケールアウトを実現できます。もし負荷が増えたとしてもサーバを増やせば負荷分散できるためサーバの能力が落ちることはなく、一つ一つの能力も高いため高いスケーラビリティが得られるのである。

次に高可用性であるが、上記でも述べたようにスケールアウトが実現できるのでデータを分散させて保存しているため１台のサーバが停止しても他のサーバにあるデータにアクセスできるため高可用性を得られるのである。

3.2Google　App　Engine

3.2.1 Google　App　Engine とは

Google　App　Engine とは、米 Google 社が提供しているクラウドコンピューティングサービスにおいて述べた 3 つの形式の内の 1 つである PaaS 形式で Web アプリケーションプラットフォームである。これは、Google 社が提供する検索サービスを構築する際に使用している様々な技術を、Google 社外の人々に利用できるようにまとめたものである。

その技術の例としては、Google 社が開発したファイルシステムである「Google　File System」や、Google の様々サービスに利用されているデータ管理機能である「Bigtable」などがある。このように Google 社が提供しているサービスと同じインフラを Google　App Engine で利用しているということは、Google 社が運営しているサーバ郡を一般ユーザーが利用できるのと同等である。

そして、Google　App　Engine を利用したアプリケーションを開発するためには、米 IBM 社が開発した統合開発環境である「Eclipse」が最も広く利用されている。Eclipse には、Google　App　Engine を開発するために必要なプラグインソフトである「Google　Plugin」が用意されており、これを Eclipse に導入するだけで、すぐに Google　App　Engine を利用したアプリケーションを開発できる。

Google　App　Engine は、2008 年にスクリプト言語「Python」に対応したクラウドコンピューティングサービスとしてリリースされ、2009 年には、Java もサポートされた[2]。

本研究では、Java 版の Google　App　Engine を使用する。

（日経BP，DVD，佐藤一憲，2010）

Google　App　Engine の Stack の構成である。

3.2.2Google　App　Engine のメリット

他のクラウドコンピューティングサービスと比較して Google　App　Engine を利用するメリットだが、以下のような点が挙げられる。

・開発環境のコストが低い
・サーバの増強が自動で可能
・サーバ運用技術が不要

「開発環境のコストが低い」というのは、Google　App　Engine は無償のソフトだけで開発環境が整えられるからである。

Google　App　Engine 上で動作するアプリケーションの開発に必要なのは、プログラム本体を編集しビルドするのに必要な統合開発環境である Eclipse、開発キットである「App Engine　SDK（Software　Development　Kit)」などである。これらは、無料で提供されている[2]。

次に「サーバの増強が自動で可能」である。例えば、クラコンピューティングサービスを一つである Amazon　EC2 は、1 台のサーバ機を仮想化技術によって複数台の仮想マシンによって分割して提供している。このような場合、サーバを追加する際には、人手による作業が必要になる。

しかし、Google　App　Engine では、サーバへのアクセス数を検知し、サーバに割り当てる CPU 能力やメモリー量、ディスク量を必要に応じて増やす。アプリケーション開発者は、Google　App　Engine の制約を守ってプログラムの作成を行えば、サーバの能力を必要な分だけ増やしていく「スケールアウト」のための設計を意識的に組み込む必要がなく必要に応じてサーバの能力の増強を行う。仮に、サービスへのアクセス数が爆発的に増加しても、システムがダウンすることは起きない。

「サーバ運用技術が不要」ということだが Amazon　EC2 をはじめとする IaaS を利用したサーバでの運用方法、Web サーバやデータベースサーバなどを各種サーバの監視やログ管理を利用者側がする必要があり、サーバで動作しているサービスのセキュリティ情報などにも注意を払い、対策が必要になる。このため、サーバ運用の技術が要求されるのである。

これに対して、Google　App　Engine では、Web サーバなどの基本的なサーバ運用、サーバに対するセキュリティ対策などは、Google 社が行なっている。さらに、このコストはサーバの利用料に含まれている。

Google　App　Engine を利用すれば、開発者は、サーバ運用やメンテナンスなどに時間を費やすこともなく、サービスの開発に時間を充てることができる[3]。

3.3Bigtable と Datastore

Bigtable は、Google 社によって Google File System や Chubby Lock Service、その他のいくつかの Google 社のプログラムで構築・開発されたキーバリュー型データベースの巨大分散データストアである。膨大な数の汎用サーバを繋げてペタバイト規模のデータを扱えるように設計されている。Google 検索をはじめ、YouTube や Google マップ、Google Earth、Google Analytics など様々な Google のアプリケーションの基盤として利用されている。

特徴としては、「無制限のスケーラビリティ」と「サーバ冗長化による高可用性」の２つが挙げられる。「無制限のスケーラビリティ」とは、Bigtable のテーブルの規模が 100 件でも数千万件でも、個々の行の読み書きは数 10 ミリ秒程度で完了する。また、膨大な数のユーザーが Bigtable にアクセスしてもレスポンスの低下が発生しない。一般的な Web アプリケーションの大半では、RDB サーバがボトルネックとなってデータ量やアクセス件数の増加にともない、レスポンス時間が低下するが、Bigtable を使った Web アプリケーションではない。次に「サーバ冗長化による高可用性」だが、データは Google 社独自の分散ファイルシステムである Google　File　System によって異なるラックに設置された３台以上のサーバにコピーされるためサーバ障害によってデータが失われる可能性は極めて低く、１台のサーバが停止しても他の２台のサーバのいずれかから同じデータを瞬時に取得する。また、Bigtable のサービスを構成するサーバ郡はすべてが冗長化されており、SPoF（Signal Point　of　Failure）を排除するため、こういった冗長構成のため Oracle　RAC（Real Application　Cluster）などのハイエンドの RDB クラスタに匹敵する高可用性をそなえている。

Bigtable は、個々の行に割り当てられたキーを指定して行の CRUD（追加、取得更新、削除）を行う「キーに基づく行の CRUD」とキーの前方一致検索もしくは範囲指定検索により、複数の行を一括取得する「キーに基づくスキャン」の２つが出来る。行の CRUD に際しては、ACID 特性を保証されているが複数行にわたっての ACID 特性は確保されていない[3]。そして、その Bgitable を一般ユーザーが利用できるようにしたものが Datastore である。

Datastore とは、Google 社の Bigtable 上で実装されており、その Bigtable をアプリケーションから汎用的に利用できるようにするサービスのことである。Bigtable にはない機能である「データの値を条件にした検索」、「複数エンティティをまたがるトランザクション処理」という機能を加えている[3]。

Bigtable の場合、キーを指定して対象データを取り出すが、Datastore では、自動的にデータをキーに使うインデックスを別途生成し、それを使ってデータで検索を可能にしている。また、複数エンティティをグループ化してトランザクション処理を実行できる。

4.どこでも画像アップロード

4.1 システムの設計

4.1.1 システムの仕様

本研究では、画像を組込型データベースの SQLite を利用したアプリケーションからキーバリュー型データベースの Datastore に保存し、Web ブラウザ上から保存した画像を見ることができるシステムを試作した。

本システムは、SQLite を利用した松江の飲食店のデータベースのアプリケーションで飲食店を選択し、コメントを入力しアップロードしたい画像を選択する。コメントと画像を保存と同期を行うか選択する。保存と同期を行う場合は、コメントに関しては画像の URI と共に SQLite に保存され、画像はサーバと同期を行う。保存と同期を行わない場合は、終了する。実際に保存された画像は、画像を見るために用意したサイトから見ることができる。

4.1.2 製品仕様

統合開発環境である Eclipse を使用してパソコン上で Android のシミュレータを実行しシステムを利用する。以下に、開発環境を示す。

表 4.1　クライアント開発環境

クライアント用機種	Android2.3.3
システム搭載サーバ	Windows7
システム実行用シミュレータ	eclipse-java-galileo-SR2-win32
Java のソフトウェア開発キット	Java SE Development Kit 6.0
Android のソフトウェア開発キット	Android SDK 1.5
Android アプリケーションを開発するためのプラグイン	ADT Plugin for Eclipse

表 4.2　サーバ開発環境

システム搭載サーバ	Windows7
システム実行用シミュレータ	eclipse-java-galileo-SR2-win32
Java のソフトウェア開発キット	Java SE Development Kit 6.0
Google App Engine の開発用ライブラリ	Google App Engine SDK for Java
Google App Engine アプリケーションを開発するためのプラグイン	Google Plugin for Eclipse

注）

Android SDK 1.5

Android のアプリケーションを開発するためのツール群。

ADT Plugin for Eclipse

Android のアプリケーションを開発するための Eclipse 用のプラグイン(機能拡張ソフト)。

Google App Engine SDK for Java

Google App Engine で Java アプリケーションを開発するためのツール群やライブラリ群である。Google App Engine 向けのアプリケーションを自分のパソコンで実行するためのツール類、Google App Engine アプリケーションに必要なライブラリ、Google App Engine に搭載された機能を利用したサンプルアプリケーションなどが含まれている。

Google　Plugin　for　Eclipse

Google　App　Engine のアプリケーションを開発するための Eclipse 用プラグイン。プラグインをインストールすることでデバッグや実行などの Eclipse の機能を統合させた上で、Google　App　Engine のアプリケーションを開発できる。

表 4.3　クライアント側データベースのテーブル SHOP

主キー：_id

列名	型	内容
_id	INTEGER	店舗の ID
NAME	TEXT	店舗の名前
ADDRESS1	TEXT	店舗の都道府県
ADDRESS2	TEXT	店舗の市区町村
ADDRESS3	TEXT	店舗の番地
TEL	TEXT	店舗の番号

表 4.4　クライアント側データベースのテーブル MEMO

主キー：_id

列名	型	内容
_id	INTEGER	コメントの ID
SHOP_ID	INTEGER	店舗の ID
MEMO	TEXT	コメント
IMAGEURI	TEXT	画像の URI
IMAGE	BLOB	画像
AT	TEXT	コメントの保存時間

表 4.5　サーバ側データベーステーブル

主キー：imagekey

列名	型	内容
Imagekey	Key	データベースのキー
shopId	String	店舗の ID
Imageuri	String	画像の URI
Image	Blob	画像

4.2 システムの内容

4.2.1 概要

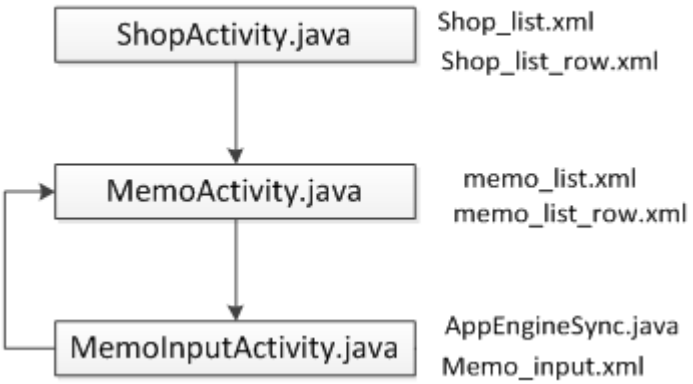

図 4.1

ShopActivity.java：店舗画面

あらかじめ作成したデータベースから店舗のデータを取得。

Shop_list.xml：ShopActivity.java の構成

ShopActivity で取得したすべての店舗のデータを表示。

Shop_list_row.xml：Shop_list.xml の構成

ShopActivity で取得した各店舗の詳しいデータを表示。

MemoActivity.java：店舗メモ画面

あらかじめ作成したデータベースに格納されている各店舗のコメントを取得。

Memo_list.xml： MemoActivity.java の構成

MemoActivity で取得したすべてのコメントを表示。

Memo_list_row.xml： Memo_list.xml の構成

MemoActivity で取得した各コメントの詳しいデータを表示。

MemoInputActivity.java：コメント入力画面

コメントの入力や画像の選択を行い保存する。

Memo_input.xml：MemoInputActivity.java の構成

コメント入力のためのエディタや画像選択のアイコンを表示。

AppEngineSync.java：Datastore との同期

クライアントとサーバを連携させる。

連携を行う方法は、まず、MemoInputActivity.java において ContentProvider と呼ばれる Android のアプリケーションフレームワークを利用して、画像が保存されている別のアプリケーションにアクセスを行い、画像を取得する。また、アプリケーションには URI と呼ばれるがありそれを使用して画像を取得している。

そして、取得した画像は、JSON と呼ばれるデータ交換用のフォーマットを利用して画像を Datastore と同期し画像をサーバ側へ送信する。

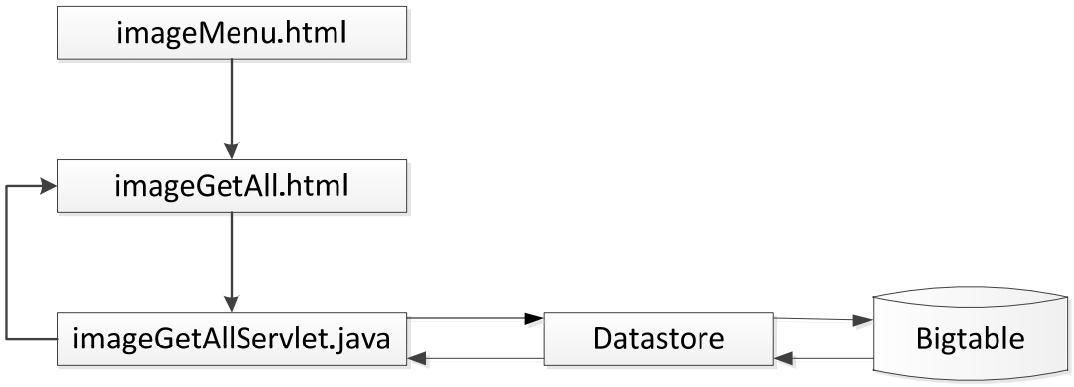

図 4.2

imageMenu.html：トップページ

どこでも画像アップロードを Web ブラウザ上に表示するためのブラウザのトップページ。

imageGetAll.html：画像全件表示画面

Datastore に保存された画像を Web ブラウザ上に表示する。

imageGetAllServlet.java：画像取得

Datastore に格納された画像を取得する。

4.2.2 クライアント側

クライアント側では､XML によってアプリケーションの構成を設定し ConteontProvider を利用して画像を共有し、JSON を利用してサーバにデータを送信している。

まず、XML とは、テキスト形式のデータフォーマットで汎用的なデータ交換用に策定されたものである。また、マークアップ言語であり、タグでデータを挟んでデータの意味を定義します。同じマークアップ言語である HTML と違い開発者が独自にタグを定義できるのである[2]。このタグを階層的に定義すれば、例えば「店」というデータに対しては、「住所」や「電話番号」といった詳しいデータがあるというふうにデータを階層構造に表現できる。さらに､多くのプログラミング言語に XML を扱うライブラリが用意されているため、言語やプラットフォームを問わずに利用できるフォーマットになっている。

次に、ContentProvider とは、データを保存・格納しアプリケーション内のデータを他のアプリケーションからアクセスできるようにする仕組みである。データのアクセスには ContentURI と呼ばれるアプリケーションやアプリケーション内のデータを識別するための識別子を利用してデータを共有する。

JSON とは、ノ―テーション(表記法、記し方)の一種で、上記で説明した XML と同様にテキストベースのデータフォーマットとして使用できる。また、JSON は JavaScript の文法の一部を使って定義しています。そのため JavaScript と親和性が高く、Ajax を利用する際のデータフォーマットとして最も多く使われている。現在では、JavaScript 以外の言語でも広く適用されるようになり、様々な情報機器のデータ交換用フォーマットとして利用されている[2]。本システムではクライアントとサーバとの間でのデータ交換のフォーマットとして利用している。

4.2.3 サーバ側

サーバ側では、JavaScript のライブラリである jQuery と JSP を利用して Web アプリケーションを作成し、JDO を利用して Datastore にアクセスします。

JavaScript とは、Web アプリケーションを作成するためのスクリプト言語である。Sun 社の Java 言語と似た記法を用いるが互換性はない。そして、この JavaScript のライブラリの一つが jQuery である。jQuery は、Ajax による通信も容易に実装でき、頻繁に利用する機能を数行で実現できるためプログラムを短くできるというメリットがあります[2]。

次に、JSP とは Java 言語を利用して Web サーバで動的に Web ページを生成し、Web ブラウザに送信する技術である。HTML ファイルの中に Java プログラムを埋め込んでおき、Web ブラウザの要求に応じてプログラムを実行、処理結果のみをクライアントに送信する。結果は通常の HTML 形式になるため、Web ブラウザに特殊な機能を組みこむことなく Web アプリケーションを構築できる[2]。これらは、主に画像を表示する HTML 内で利用する。

JDO とは、データが含まれるオブジェクトをデータベースに格納するためのインターフェースである。サーバ側の Google　App　Engine の Datastore にアクセスしてデータを保存あるいは習得するために利用する。

4.2.4 実行画面

あらかじめ SQLite を使用して作成した松江の飲食店のデータベースを図 4.3 のように表示し、自分がコメントや画像を追加したい店舗を選択する。次に、図 4.3 において選択した店舗の情報やあらかじめその店舗についてのコメントを図 4.4 のように表示する。その店舗についてコメントや画像を追加する場合は、「コメントを追加」ボタンを押し、コメントと画像を追加する。追加しない場合は、店舗の選択画面の図 4.3 に戻る。

次に、コメントと画像を追加するための画面を図 4.5 のように表示する。エディタによってコメントを編集し、「画像」ボタンを押すことで画像を選択する画面へ移動できる。そして、図 4.5 において「画像」ボタンを押した場合は、Android に入っている画像が図 4.6 のように表示され追加したい画像を選択することが出来る。追加したい画像がある場合は、画像を選択し図 4.7 のように画面が変わり、無い場合は図 4.5 に戻る。画像を選択した場合は、コメントと一緒に選択した画像が図 4.7 のように表示される。コメントと画像を追加する場合は、「保存と同期」ボタンを押しコメントはデータベースに保存され、画像はサーバと同期し Datastore に保存される。「キャンセル」ボタンを押した場合、図 4.4 に戻る。保存と同期は、コメントだけ保存することや画像だけ同期することも可能である。

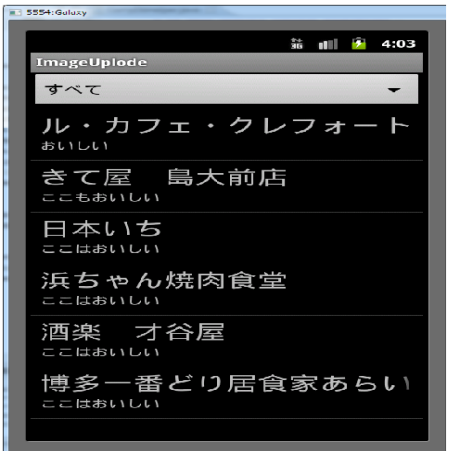

図 4.3

図 4.4

図 4.5

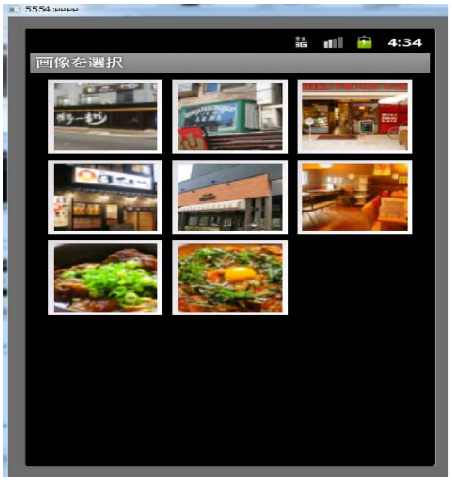

図 4.6

図 4.7

アプリケーションによって同期した画像は Web ブラウザによって見ることが出来る。Datastore に保存された画像を見るための Web ブラウザのトップページが図 4.8 である。図 4.8 において「イメージ全件表示」を押すことでアプリケーションによって同期して Datastore に保存された画像がすべて図 4.9 のように表示され、画像を見ることができる。

図 4.8

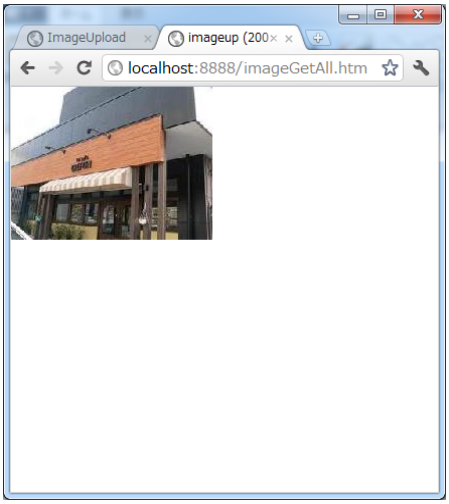

図 4.9

６.結論

本研究の目的である組込型データベースとキーバリュー型データベースを連携させ、どこでも画像アップロードを試作することができた。今後の課題としては、Web 上からは誰でも画像が見ることが出来てしまうのでパスワードの設定をすることと、サーバからクライアント側へ画像を送信できるようにすることである。

謝辞

本研究において最後まで熱心な御指導をいただきました田中章司郎教授には心より御礼申し上げます。また、同じ研究室のみなさんには、数々の御協力と御助言を頂きました。厚く御礼申し上げます。

本論文、本研究で作成したプログラム及びデータ、並びに関連する発表資料等の全ての知的財産権を本研究の指導教官である田中章司郎教授に譲渡致します。

引用文献・ホームページ・資料

[1]日経ソフトウェア 2011.9
[2]クラウド活用のための Andoird 業務アプリ開発入門　著　出村成和
[3]クラウドコンピューティング設計・開発　Google App Engine 編　著　佐藤一憲