一般社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
MoMuC2012-30(2012-9)

Androidアプリケーション開発におけるデータベース構築手法の検討

佐々木 喜一郎† 安田 孝美‡

†岐阜経済大学経営学部情報メディア学科 〒503-8550 岐阜県大垣市北方町 5-50
‡名古屋大学大学院情報科学研究科 〒464-8601 名古屋市千種区不老町
E-mail: †sasaki@kiichiro.jp, ‡takami@nagoya-u.jp

あらまし 現在，Android アプリケーションのデータベース構築は，SQLite のライブラリを活用した手法が主流である．ゆえに，仕様以上のデータ容量や構造によるデータ処理速度の低下が著しい事から，高速でデータ処理が必要な Android アプリケーションに不向きである．また，データベースにパスワードが設定出来ない事から，機密データを扱う Android アプリケーションにも不向きである．本研究では，Android アプリケーションの特性に合わせたデータベースを構築する手法について，様々なデータ処理シミュレーションにより比較検討する事で，多様な Android アプリケーションを実現できる可能性を示す．

キーワード Android，データベース，SQLite，シミュレーション

Examination of Database Building Method for Android Application Development

Kiichiro SASAKI† Takami YASUDA‡

†Faculty of Business Administration, Department of Information and Media Studies, Gifu Keizai University
5-50 Kitagata-cho Ogaki-shi, Gifu Prefecture, Japan 503-8550

‡Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, Nagoya, Japan 464-8601
E-mail: †sasaki@kiichiro.jp, ‡takami@nagoya-u.jp

Abstract Nowadays, the database construction in Android application is mainly using the library of SQLite. Therefore, because of the data capacity over the specification and the decreasing in the data operation speed by the construction are not suitable for Android application which needs a high speed operation. Also the database could not set any passwords, so it is not suitable for Android application which treats secret data. In this research, by comparing and considering various data operating simulation about the technique of constructing database fits to the characteristic of Android application. We indicate the possibility of realizing many kinds of Android application.

Keyword Android, Database, SQLite, Simulation

1. はじめに

スマートフォンアプリケーション市場は，2011 年において 82.2 億円の規模であり，2012 年には 139.9 億円前年比 170%の規模になると予測され，依然として市場が成長している．このような状況の中，新しい Android 端末の仕様が日々変化している為，開発手法についても模索段階である．

本研究室は，産学連携プロジェクトにおける Android アプリケーション開発として，ヘイ！タクシー！を開発した．この Android アプリケーションは，自分自身の場所を GPS で探知し，その場所から近い距離のタクシー会社の場所を，全国タクシー会社の情報で構成されているデータベースから検索し，電話番号等の情報を提供する．当初，全国タクシー会社のデータベースは，クラウド上のデータベースサーバーを活用する事により，十分な反応速度を得られていた（図 1）．しかし，ヘイ！タクシー！は，クラウド環境が利用不可能な場所にも対応して欲しいという要望が多く，Android 端末内にデータベースを内包する方式を採用する事により要望に応えた（図 2）．しかし，以前の方式と比べて，情報を提示する速度が大きく低下する課題が浮き

彫りになった．しかし，一般的な SQLite[1]ライブラリを活用した手法は，２つの問題がある．SQLite ライブラリを改良する事は，全体的な構造を見据えた改良が必要な為，非常に困難である事が挙げられる．また，ライセンスによる制限は無いが，改良可能な範囲が限られている．故に，汎用性が高い枠組みであるアプリケーションフレームワークを活用した手法に注目した（図 3）．本論文では，アプリケーションフレームワークを基盤としたデータベース構築手法及び活用方法について検討し，その有効性を示す．

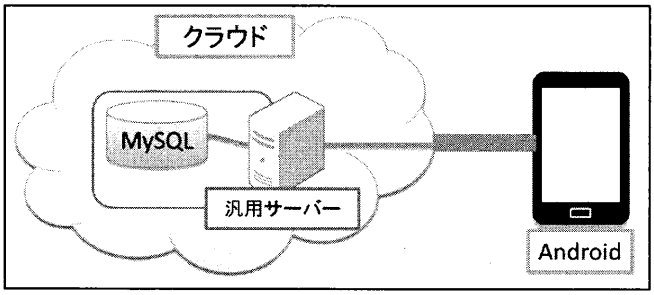

図 1．クラウド DBMS 方式

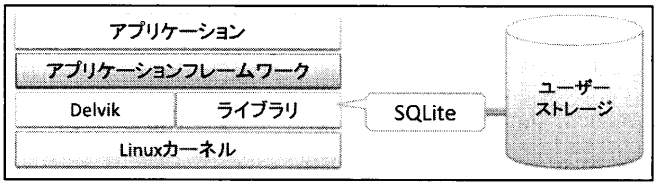

図 2．SQLite 方式

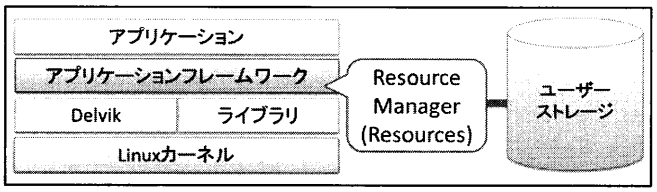

図 3．本提案方式

2. 関連研究

本研究に関する研究は，下記の３点に分類できる．Android におけるデータベースの構築手法，スマートフォンに対応したクラウド DBMS の効率的な運用方法，スマートフォンに対応したクラウド DBMS の活用方法である．

木島らは，Android アプリケーションのデータベースを構築する場合に SQLite ライブラリを利用する方法を提示している[1]．片山は，Android プログラミングの実践方法として，SQLite ライブラリを利用したデータベース構築方法を提示している[2]．間は，Android のデータベース設計について，SQLite ライブラリを基礎とした方法を提示している[3]．これらの論文及び書籍では，Android アプリケーションのデータベースを構築する場，SQLite を利用する事を前提としており，効率的なデータベースの構築方法や運用方法について言及されていない．

太田らは，スマートフォンの多様な操作履歴データを活用するためのミドルウェア向けに，携帯機のリソース制約に対応した 2 つのプライバシ保護方式を提案する際に，クラウド DBMS の運用方法によるデータ収集方法を提示している[4]．しかし，スマートフォンに対応した DBMS の効率的な運用方法について言及されていない．

北島らは，PDA やスマートフォンを対象にしたデータの処理方法について，クライアントの消費電力を考慮した方式について提案している[5]．渡辺らは，スマートフォンを対象として，ユーザが現在閲覧中のウェブページと関連した情報を検索することを支援する仕組みにおいて，クラウド DBMS を活用している[6]．これらの論文では，クラウド DBMS を活用した定量的な評価実験及び分析がなされていない．

3. SQLite 方式

3.1. 提案方式の概要（ブロック図）

SQLite 方式（図 4）は，データベースに接続して利用する為，SQLiteOpenHelper を継承したクラスが必要である．また，スマートフォンの画面制御 Activity クラスが必須となる．DBHelper は，SQLiteOpenHelper を継承したクラスであり，必須メソッドのみの構成としている．Activity は，DBHelper クラスのオブジェクトを作成，そのオブジェクトを用いたデータ入力及びデータ抽出をする．データ処理の詳細を 3.2.に示す．

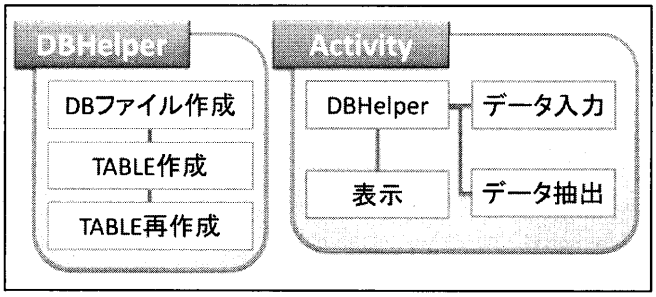

図 4．SQLite 方式のブロック図

3.2. 提案方式の動作手順（フローチャート図）

SQLite 方式（図 5）は Activity クラス内において関数化した処理を示す．SQLite 方式はデータベースファイルをアプリケーション内で作成して動作する．

まず DBHelper クラスのオブジェクトを利用し，データベースファイルを作成する．作成したデータベースファイルにデータの挿入をレコード数まで繰り返し処理をしている．データの抽出は cursor を利用してレ

コード抽出を行う．返り値として配列に格納し，onCreate メソッドに値を返す．

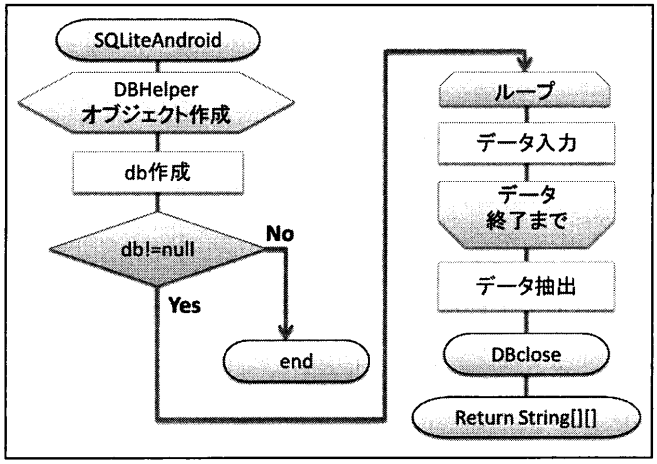

図 5．SQLite 方式のフローチャート図

3.3. SQLite 方式の動作

バッチ処理により，Android エミュレーターを起動し，Android アプリケーションをインストール及び実行した動作状況である（図 6）．出力結果は，データベースの構築結果の一部を表示している．

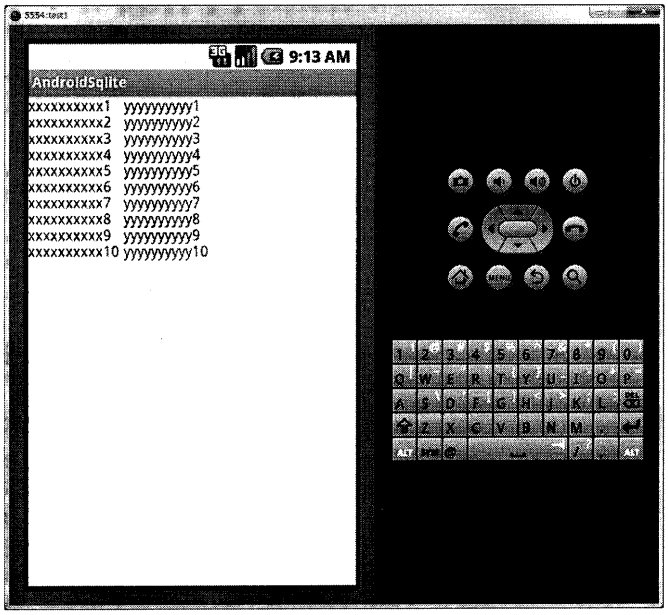

図 6．SQLite 方式の動作画面

4. 提案方式

4.1. 提案方式の概要（ブロック図）

本提案方式では，アプリケーションフレームワークの ResorceManager を利用した方法を提案する．大量のデータをオフライン状態で扱う事を想定し，初期動作をいかに高速化するかを検討した．

機能として今回は，読み込み，カウント，分割，抽出に制限した（図 7）．なお，アプリケーションのデータベース利用方法により機能の追加は可能である．データの処理の詳細を 4.2.に示す．

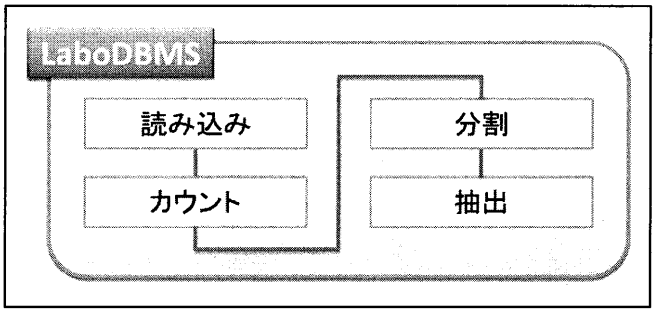

図 7．提案方式のブロック図

4.2. 提案方式の動作手順（フローチャート図）

本提案方式（図 8）の動作手順においてアプリケーションパッケージ内のリソースフォルダにデータファイルを格納しておく必要がある．なお，データファイルはあらかじめ決められた文字で区切られたデータ群である．

まず，カウント用変数においてレコード数を把握する為のもの，レコード位置を確認する為のもの，フィールド位置を確認するものの 3 つを宣言する．

Resources クラスのオブジェクトを作成し，リソースフォルダのデータファイルにアクセスする．InputStreamReader クラスのオブジェクトを作成する際，Resource クラスのオブジェクトを利用する事で，ファイルの読み込む準備を行う．BufferedReader クラスのオブジェクトを作成するにあたり，InputStreamReader クラスのオブジェクトを引数にする事でデータファイルをレコードとして読み込むことが可能になる．

レコードの終了までカウントしながら繰り返し処理をする事でレコード数を把握する．

レコードをフィールドに分割する為にレコードを一次的に保存する変数と抽出後のデータを格納する配列を宣言する．なお，配列の要素数はレコードの総数で確保する．レコードをフィールド毎に読み込む為には，StringTokenizer クラスを利用する．StringTokenizer のオブジェクトを作成する際にレコードとして読み込んだ値を第一引数，決められた区切り文字を第二引数にする事でレコードをフィールド毎に読み込み可能にする事が出来る．

フィールドの位置が必要なフィールドであるかを判別し，必要ならば配列に格納し，不必要ならば次のフィールドを読み込む処理を繰り返すことで全てのレコードにおいて，同じフィールドの位置にあたるデータを抽出する事が出来る．抽出したデータは配列で onCreate メソッドに値を返す．

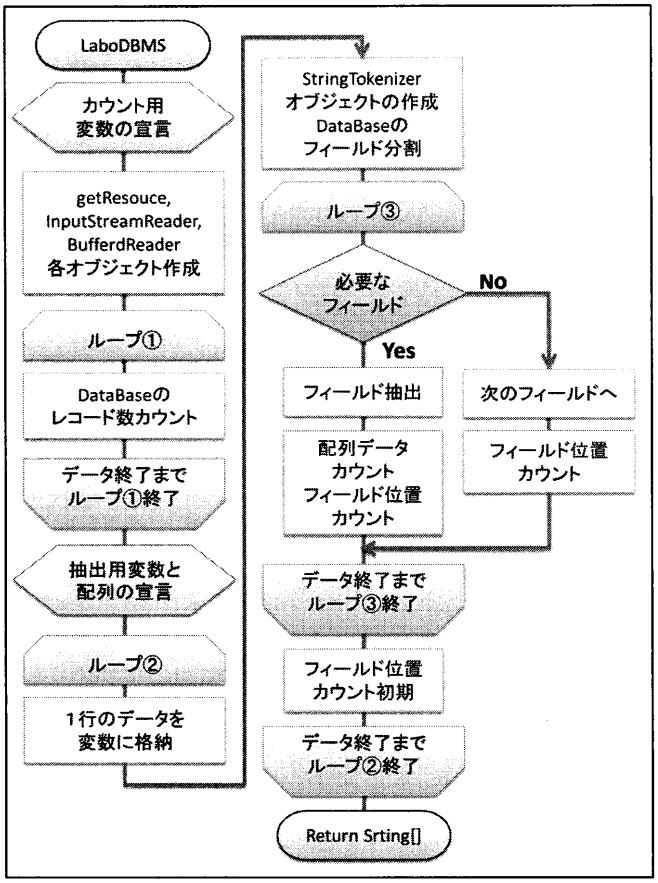

図 8. 提案方式のフローチャート図

4.3. 提案方式の動作事例(キャプチャ画像)

バッチ処理により，Android エミュレーターを起動し，Android アプリケーションをインストール及び実行した動作状況である（図 9）．出力結果は，データベースの構築結果の一部を表示している．

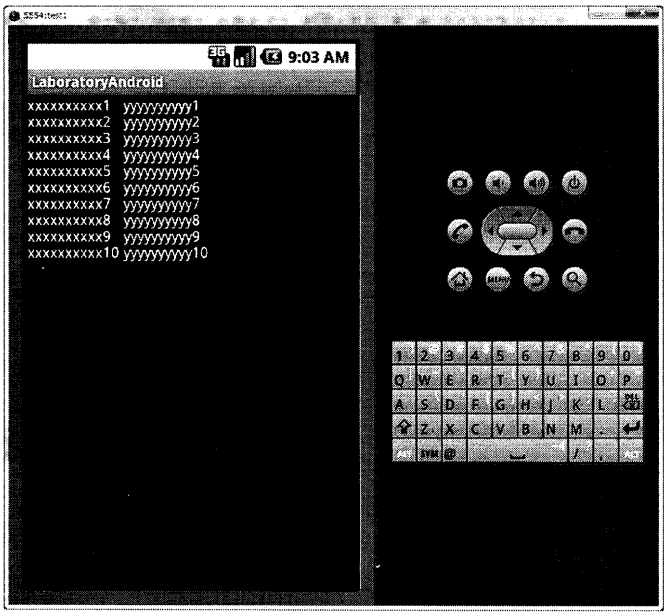

図 9. 本提案方式の動作画面

5. 各方式の比較実験

5.1. 実験方法

SQLite 方式と提案方式の効率性を比較する為，(表 1) の環境下において，下記のバッチ処理を実行し，Android Trace View[8]により実験データを収集した．また，データベース処理は，フィールドを 10 単位で固定し，レコードを 500 単位別毎に追加して実験を実施した．

バッチファイル JikkenMuster.bat を作成し，android-sdk-windows/tools をあらかじめ，システム環境変数のパスを通す．start emulator -avd [Android エミュレーター名]により，別コマンドプロンプトにて選択した Android エミュレーターを起動する．ping localhost -n 120 > null により，android emulator が完全に起動するまで 120 秒程待機する．

cd c:¥android-sdk-windows¥platform-tools により，起動した Android エミュレーターに送るコマンドを実行する為，c:¥android-sdk-windows¥platform-tools に移動する．adb uninstall [削除するパッケージ名] により，adb コマンドで起動した Android エミュレーターにデータを送信する．uninstall [削除するパッケージ名]において，Android エミュレーター内の削除するパッケージを選択し削除する．ping localhost -n 20 > null により，パッケージが削除完了まで 20 秒待機する．adb install フルパス¥[ファイル名].apk により，windows 内の Android パッケージファイルを Android エミュレーター内にインストールする．ping localhost -n 20 > null により，インストールが完了するまで 20 秒程待機する．adb shell < フルパス¥[ファイル名].txt により，アプリケーションを自動的に起動させる（図 10）．

ping localhost -n 120 > null により，実行が完了するまで 120 秒程待機する．adb pull /フルパス/[ファイル名].trace 保存先により，android emulator 内に保存されているアプリケーションの実行結果ファイル/フルパス/[ファイル名].trace を windows 側の保存先へ出力する．taskkill /im emulator-arm.exe により，android emulator を終了する．終了するまで 20 秒程待機する．

以上のバッチ処理を，下記の 15 種類の Android エミュレーターで実行する事により，Android エミュレーター毎の特性データを収集した．使用した Android エミュレーターは，GoogleAPIs プラットフォーム 2.1, GoogleAPIs プラットフォーム 2.2, GoogleAPIs プラットフォーム 2.3.3, GoogleAPIs プラットフォーム 3.0, GoogleAPIs プラットフォーム 3.1, GoogleAPIs プラットフォーム 3.2, GoogleAPIs プラットフォーム 4.0, GoogleAPIs プラットフォーム 4.0.3, GoogleAPIs プラットフォーム 4.1, DTS Add-On 京セラ プラットフォーム 2.2, Real3D Add-On-LGE プラットフォーム 2.2,

GALAXY Tab-Samsung プラットフォーム 2.2, Intel Atom x86 System Image-Intel プラットフォーム 2.3.3, EDK2.0-sony プラットフォーム 2.3.3, ICS_R2-Motorola プラットフォーム 4.0.3 である.

表 1. 実験環境

CPU	Intel(R)Core(TM)2 Q9550 2.83GHz
Memory	DDR2 PC2-6400 4.00GB
OS	Windows7Professional 64bit

```
am start -n package/package.[ Activity ファイル名]
exit
```

図 10. Android エミュレーターシェルスクリプト

5.2. 実験結果

前節の実験方法により収集したデータを集計した. この集計項目の意味として, inclCpuTime は関係メソッド総処理時間を表し, ExclCpuTime はメソッド単体処理時間を表し, msec は総処理時間を表す. (表 2) (表 3) は, 15 種類の Android エミュレーターによる実験結果の平均値である. この結果について, 比較検討しやすくする為, グラフ化した (図 11) (図 12) (図 13). これらの各実験結果を分析した結果, 本提案方式における以下の特性を発見した.

本提案方式は, SQLite 方式と比較して, レコード数に対するメソッド数が大幅に削減されており, メモリの使用量が低減された. また, 1500 レコードまでの処理時間が高速である事が明らかとなった. しかし, メソッド単体の処理に時間がかかる事から, 同時処理スレッド数が少なく, 単体の周波数が高い CPU に適していると判明した. 結果, ミドルクラスまでの Android 端末に適しており, 汎用性が高い Android アプリケーションを開発し, 提供する事に適した特性を有していることが判明した. これにより, データベース内蔵方式における 1500 件までのデータを取り扱う際に極めて効果的である事が判明した.

表 2. SQLite 方式の実験結果

SQLite	method	inclCpuTime	ExclCpuTime	msec
500	1056	2055.40	7.60	2382.67
1000	960	2297.36	8.85	2382.00
1500	960	2468.64	10.38	2420.00
2000	946	2166.89	44.10	2448.00
2500	946	2171.05	50.80	2495.33
3000	947	2199.56	82.35	2509.33

表 3. 本提案方式の実験結果

本方式	method	inclCpuTime	ExclCpuTime	msec
500	184	1085.47	321.53	1227.16
1000	177	1648.19	558.27	1978.12
1500	174	1928.57	633.46	2202.72
2000	172	2091.08	654.15	2213.24
2500	176	2110.07	646.50	2238.55
3000	177	2098.71	651.28	2206.75

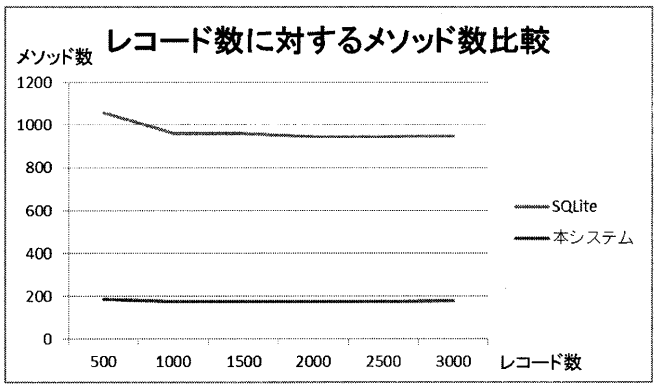

図 11. レコード数に対するメソッド数比較グラフ

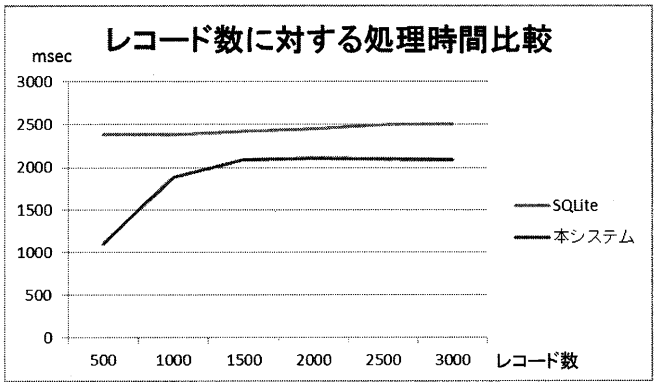

図 12. レコード数に対する処理時間の比較グラフ

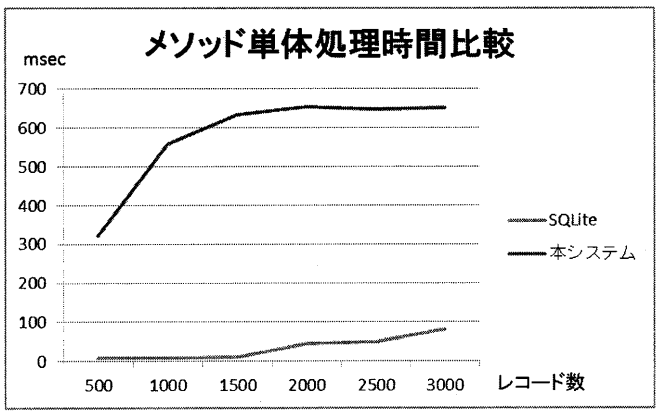

図 13. メソッド単体処理時間の比較グラフ

6. 方式の判定

我々は，5.2.実験結果から，Android アプリケーション開発におけるデータベース構築方式の判定（図 14）を次に示す．オンライン環境が利用可能な場合は，クラウド DBMS を推奨する．オフライン環境の利用を前提とした場合は，いくつかの条件により判断が分かれる．インメモリデータベースを基盤とする場合やレコード数が 3000 件以下かつ,データファイルを最小限にする事を考慮しない場合は，SQLite を推奨する．レコード数が 1500 件以下の場合やレコード数が大量かつデータファイルを最小限に抑える場合は，本提案方式を推奨する．

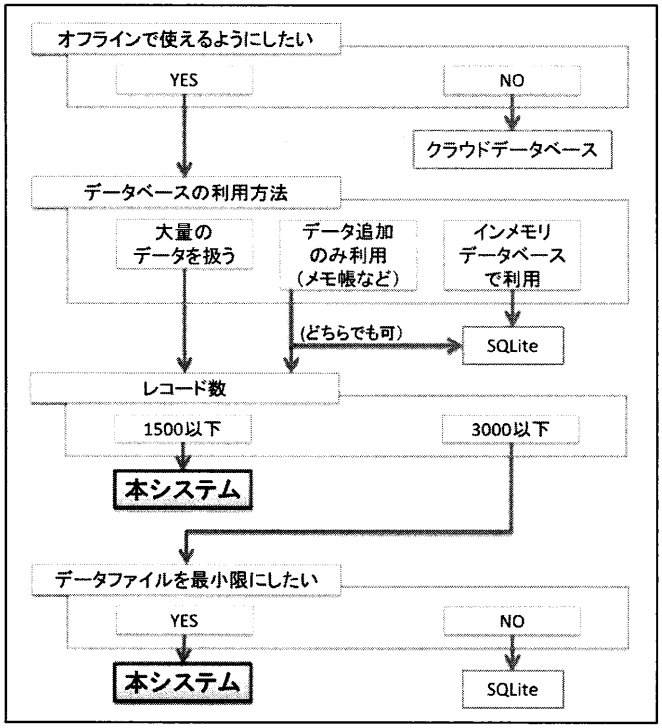

図 14. データベース構築方式の判定図

7. まとめ

本研究では，Android アプリケーションの開発にあたり，データベースの構築及び，利用する方式について検討した．結果，データベースの構築する環境や条件により，判断が異なる事を実証実験から明らかにした．また，本研究は新しいデータベース構築及び利用する方式について，一定の条件と利用環境において，有用的であると証明した．この提案方式により，Android アプリケーション開発におけるデータベースの構築及び利用に選択肢を得られた．

今後，Android アプリケーションの開発手法について様々な手法が生じる大きな可能性があり，本提案方式の新たな応用方法もあると考察した．

8. おわりに

実験方法について，より詳細な実験結果を得る為，レコードを 500 単位毎に処理する方法から，レコード単位を細かく処理する方法にしたい．これにより，より厳密な判断を行う方法や数式を得られる予定である．

また，今回の実験は，データベースの読み込みと抽出に着眼しており，データベースの書き込みについても検討したい．

さらに，実験環境は Android エミュレーターにより実施している為,各種実機を用いた実験環境を整備し，より現実的なデータを収集及び分析したい．

その他に，本提案方式をマルチスレッドに対応させる事により，処理時間をさらに短縮させ，汎用性が高い方式を実現したい．

以上を実現する事により，Android アプリケーション開発者に有益な情報を提供したい．

謝辞

本研究を進めるにあたり,岐阜経済大学情報技術研究所ソフトピア共同研究室，岐阜県財団法人ソフトピアジャパン，株式会社量子情報の皆様には，多大なご協力を頂きました．また，実証実験にあたり，岐阜経済大学の久世裕也様には,多大なご協力を頂きました．ここに深謝いたします．

文　献

[1] SQLite, http://www.sqlite.org/

[2] 木島貴志，石丸宗平，“スマートフォンプログラミング-iPhone と Android : 2.Android プログラミング入門　Android の概要からプログラミングまで，”情報処理学会会誌，vol.52, no.4, pp.527-539, Apr.2011.

[3] 片山幸雄，Android4.0 演習―「Android アプリ」や「ゲーム」の作り方が身につく!，工学社，東京，2012.

[4] 間顕次，Android UI デザイン&データベースプログラミング，日本 Android の会 (監修)，ソシム東京，2011.

[5] 太田賢，木南克規，中川智尋，土井千章，稲村浩，“スマートフォン向けプライバシ強化型操作履歴ミドルウェアの設計と実装，” 情報処理学会論文誌，vol.53, no.2, pp.825-835, Feb.2012.

[6] 北島信哉，原隆浩，寺田努，義久智樹，西尾章治郎，“情報処理学会論文誌，vol.50, no.9, pp.2284-2297, Sep.2009.

[7] 渡辺奈夕子，岡本昌之，菊池匡晃，飯田貴之，佐々木健太，堀内健介，山崎智弘，大村寿美，服部正典，“情報処理学会論文誌，vol.50, no.9, pp.2284-2297, Sep.2009.

[8] Android Trace View, http://developer.android.com/tools/debugging/debugging-tracing.html