Amazon Web サービスを併用する センサーデータ管理システム設計

金子　邦彦*

Design of Sensor Data Management System using Amazon Web Service

Kunihiko KANEKO*

ABSTRACT

Amazon Web Services is a commercial cloud service that provides various infrastructures such as computers, storages, file processing system, database management systems, and data warehouse system. In this paper, the handling of sensor data in cloud services is discussed, and Amazon Web service is used as an example. Cloud services are not evaluated in this paper. First, regarding data cleansing, which is an important issue of sensor data, the cleansing process is the process of sorting record sets into true sets and false sets. The implementation can be done with functions and maps in programming language. As a result, parallel processing becomes easy. A set of records of the same type is handled as a file or a table so that parallel processing of the cloud system can be easily performed. Furthermore, SQL programs such as aggregation of sensor data can be performed by parallel processing in data warehouse system. As described above, cleansing and aggregation of sensor data can be performed by parallel processing on a cloud system.

キーワード：AWS, Amazon Web Service, センサーデータ, データウエアハウス, リレーショナルデータベース管理システム, グループ化, 集計集約, 並列データ処理

Keywords: AWS, Amazon Web Service, sensor data, data warehouse, relational database management system, grouping, aggregation, parallel data processing

１．はじめに

センサーデータ管理は，日々発生する大量のセンサーデータに対して，データのクレンジング，蓄積，集計集約等の処理を行うもので，高速な処理を必要とし，重要な研究領域である．一方で，種々のコンピュータ資源（コンピューティングやストレージなど）を即時に利用可能とするサービスであるクラウドサービスは，センサーデータ管理に有用である可能性がある．現在，クラウドサービスは種々多様化している．仮想的にサーバを占有するもの（ファイルシステムやプロセス等が他のユーザから隔離しているもの）や，ストレージの機能を使うなど，種々のコンピュータ資源を使えるものもある．あるいは，ファイルサーバ，データベース管理システムやデータウエアハウスシステムのサーバなど，特定のインフラを使えるものもある．従って，クラウドサービスによるシステム構築では，提供される個々のサービスの機能特性を見て組み合わせることが大切となり，単純ではない．

本来，大量のデータが日々発生するような状況では，データの蓄積に足りるストレージの容量拡張が問題

*情報工学科

となる．操作ミスでデータを失うことを防ぐための，データのバックアップも必要である．多くの商用クラウドサービスでは，ストレージの容量拡張や，データのバックアップを簡単に行える機能がある．クラウドサービスにおいても，装置の故障によるデータの棄損，不正アクセスの問題は依然として残り，別途，外部にデータをバックアップする，データを暗号化するなどの措置は必要であるものの，大量のデータが日々発生するような状況では，クラウドサービスを補完的に使うなどは十分にありえる．

本稿では，主要な商用クラウドサービスの１つである Amazon Web サービスを用いたセンサーデータ管理システムの設計を示す．2 章ではセンサーデータの取り扱いについて説明する．3 章では，Amazon Web サービスの主要サービスである EC2, S3, File Gateway, lambda, Redshift の概要を説明する．4 章では，EC2, S3, File Gateway, lambda, Redshift の組み合わせによるセンサーデータ管理システムの設計を示す．5 章では，クラウドサービスがセンサーデータ管理に役立つ状況について考察する．

２．センサーデータの取り扱い

センサーデータは，複数種類のレコード集合である．レコードは，値の並びである．センサーデータに対する処理としては，クレンジング，蓄積，グループ化と集計集約がある．以下，それぞれの概要と特質を説明する．あわせて，出発から到着までのトリップ（origin destination trip）についても説明する．

２．１　クレンジング

センサーデータは誤差を含む．処理を適切に行うため，個々のレコードデータについて，正当であるか，正当でない（大きな誤差を含む，異常は値であると判断される）かを判断するクレンジングの処理が必要となる．

誤差の要因は種々ある．センサーの故障，センサーを稼働させるソフトウエアの不具合やバグ，センサーが基礎とする仕様の解釈のゆれ，電源オン直後の動作不安定によるデータの乱れ，センサーが本質的に含む計測誤差など多数ある．クレンジングを，レコード r から {true, false} への関数 c(r) ∈ {true, false} のように見立てるとき，誤差の要因ごとに個別のクレンジング関数を考えるのが取り扱いやすい．本稿では，誤差の要因ごとに番号（この番号のことを「クレンジング番号」と呼ぶことにする）があると考えて，以下，クレンジング関数を $c_i(r)$ と書く（i はクレンジング番号）．

クレンジングは，検証が必要である．レコード集合に対して $c_i(r)$ = true であるようなレコードのみを選択した真集合と，$c_i(r)$ = false であるようなレコードのみを選択した偽集合を，コンピュータのプログラムを実行させて得る．その後，真集合と偽集合を人間が見比べて，真集合に正当なレコードが含まれ，偽集合に正当でないデータが含まれている精度等を確認し，クレンジング関数 $c_i(r)$の妥当性を検証する．必要に応じて，$c_i(r)$のプログラムを修正し再実行する．以上のような検証が必要であることから，本稿では，クレンジングは，正当でないレコードデータの除去のことではなく，レコード r から {true, false} への関数と定める．

クレンジング関数を用いて真集合や偽集合を得るとき，マップ(map)の利用が便利である．マップは，集合の全要素に対して関数を適用し結果を得るもので，多くのプログラミング言語が備える機能である．例えば，集合 S の全要素に対して関数 f を適用するマップの関数 m(S, f)は，Python プログラミング言語での pandas を使う場合，次のように定義できる．

```
import pandas as pd
def m(S, f):
  S.apply(f, axis=1)
```

クレンジング関数を，SQL で扱うときには，違う書き方になる．関数 f を，SQL のユーザ定義関数の機能により組み込んだのちに，次の SQL プログラムを実行する．A1, A2, …, An のところには，クレンジング関数のパラメータである属性名を書く．次の SQL プログラムの実行により，テーブル T の全要素に関数 f が適用される．

```
select *, f(A1, A2, …, An) from T;
```

クレンジング関数のプログラムは改版がありえ，プログラムのソースコードの版管理（バージョン番号付け）にあわせて，真集合や偽集合が，どのクレンジング関数のどの版から得られたかの管理も必要である．このことは将来の課題としたい．

２．２　データ蓄積

レコードデータの蓄積は，ファイルに行う．ファイル操作としては，バックアップ，暗号化と復号化，システム間のファイル転送がある．ファイルの中の 1 行あるいは 1 オブジェクトが 1 レコードとなり，1 ファイルの中に，複数のレコードが格納されるとする．CSV ファイルや Excel のファイルなど行を単位とするファイルもあれば，JSON のようにオブジェクトを単位とするファイルもあるため，1 行あるいは 1 オブジェクトが 1 レコードになるとしている．計測値に限

らず，センサーデータの計測日時，センサーデータの収集・蓄積日時，センサーの種類や場所なども必要に応じてデータ化し，レコードデータの中に含めることとする．

本稿では，1つのファイルの中には同種のレコードのみが格納されること，そして，ファイル内で，行やオブジェクトの順序には意味がない（ファイル内で，行やオブジェクトを並べ替えても，データの意味は変わらない）とする．このように，行やオブジェクトの順序には意味がないというように単純な格納形式を定めておくことは，データの取り扱いを容易にする．同種のレコードデータを 1 つにファイルにまとめておくか，複数ファイルに分けておくかのような，ファイル格納の指針を前もって定めることなく，必要に応じて，必要なファイルを作るという単純な運用方針で済むことになる．（なお，同一データを複数のファイルに重複格納するという冗長データの発生防止は，運用で気を付ける必要がある）．複数のファイルを1つにまとめるときでも，行やオブジェクトの順序に注意を払う必要はない．CSV ファイルの場合は，次のような単純な linux コマンドの実行などで，1 つにまとめることが可能になる．

```
cat file1 file2 > file3
```

単純な形式にしておくことは，レコードデータの並列処理を簡単に行えることにも役立つ（並列処理後，処理の結果を1つにまとめるなどのとき，元のレコードデータにおける行やオブジェクトの順序を意識する必要がなくなる）．

以上のような，単純な格納形式の場合には，ファイルの中には，メタデータ（ファイルの中身に関する説明）や属性名などを書いたヘッダ行，小計の行，コメント行などはなく，すべてレコードデータである．レコードにはさまざまな種類があるが，レコードの種類は，ファイルの名前（ファイル名）で識別されるとする．

センサーデータは日々発生し続けるが，削除，更新は基本ない．センサーデータは発生のたびにいったんコンピュータの実メモリ上に保存，ある程度まとまった時点でレコードデータのファイルに書き出すという簡単な処理方法になる．削除，更新がないため，ファイルのロック，ファイル内のレコードへの索引付けは考慮しない．

２．３　グループ化と集計集約

グループ化は，レコード集合から，その属性値に応じて複数のグループを作ることである．集計集約はクループの行数の数え上げや各グループの記述統計量（平均，中央値，４分値，最大値，最小値など）の算出を行うことである．例えば，赤と白のいずれかの値をとるレコードがあるときの，赤の値のレコード数と白の値のレコード数の算出といった処理である．

センサーデータの特質として，データの欠損がある．センサーの電源供給が絶たれている，通信が絶たれているということは頻繁に起きえる．そして，センサー自体も全域に密に配置できるとは限らない．そうした状況においては，観測網という意味ではデータの欠損がある．しかし，全センサーを，精密に全時間にわたって分析することはできなくとも，「センサーデータは，現実世界の状況を無作為に抽出した標本である」と見立てて，センサーデータから，現実世界の状況を統計的に把握する（例えば，仮説を検定したり，有意区間を定めての値の推定を行ったり）ことはあり得る．そうした状況において，グループ化と集計集約は基礎となる処理である．そして，センサーデータは大規模であることから，性能面でも注意深くシステムを構築する必要がある．

グループ化と集計集約では，全データを対象とするとは限らず，レコードの選択を行うことがありえる．センサーデータにおいても，全データを対象とするとは限らず，直近1か月の選択などがありえる．何を基準としてグループを作るかというグループ化の基準の種々変わりえる．すなわち，グループ化と集計集約の処理は定型的な処理に限定することができない．定型的な処理であれば，前もっての処理が可能であるが，非定型的な処理の場合には，処理のプログラムの作成と試験，処理のプログラムの実行という一連の作業を簡単に，素早く行える必要がある．こうしたことから，本稿では，リレーショナルデータベース管理システムやデータウエアハウスシステムの利用，SQL 言語でのプログラミングにより，非定型的な処理も簡単に素早く行えるようにすると考える．

リレーショナルデータベース管理システムやデータウエアハウスシステムでは，データの単位はテーブルである．テーブルは，同一種のレコード集合であり，テーブルにおけるレコードの順序に意味はない（数理的な議論を行うときは，「リレーション」と呼ぶこともあるが，本稿では「テーブル」という言葉で説明する）．n 個の属性リスト A1, A2, …, An によりテーブル T のグループ化と集計集約を行う処理の SQL プログラムは次のようになる．「＜式＞」のところには，SQL が定める最大，最小，合計，平均，行数カウントのキーワードである max, min, sum, avg, count を用いて式を書くことができる．

```
select A1, A2, …, An, ＜式＞ from T
group by A1, A2, …, An;
```

例えば，A1, A2, …, An による数え上げを行う場合には，次のように書くことができる．

```
select A1, A2, …, An, count(*) from T
group by A1, A2, …, An;
```

レコードの選択を行う場合には，次のようにレコードの選択のための条件を書き加える．

```
select A1, A2, …, An, ＜式＞ from T
where ＜条件＞ group by A1, A2, …, An;
```

数え上げの結果は，Python などの他のプログラミング言語で読み取り，比の算出，頻度分布を用いた統計的検定（スチューデントの t 検定，多群検定など）などの追加処理を簡単に行うことができる．

max, min, sum, avg, count で扱えないような集計集約を行うときは，SQL のユーザ定義関数の機能により新しい関数を組み込んでから上の書き方で実行するか，SQL では次のように並べ替え（ソート）のみを行い，その後，Python などの別のプログラミング言語で処理する．

```
select A1, A2, …, An, ＜式＞ from T
where ＜条件＞ group by A1, A2, …, An;
```

非定型処理を考慮すれば，以上のように，リレーショナルデータベース管理システムやデータウエアハウスシステムでの SQL の利用が有用である．このとき，2.1 で説明したファイルを，リレーショナルデータベース管理システムやデータウエアハウスシステムに取り込む（このことを「コピー（copy）」という）操作が必要である．PostgreSQL の場合には，次のようなコマンド操作で取り込むことができる．他のシステムでも多くの場合，同様の機能がある．

```
¥copy T from 1m.csv with csv;
```

２．４　その他の処理

移動する車両や人物にセンサーがあるとき，出発から到着までのトリップ（origin destination trip）を単位とした処理を行う場合がある．このとき，元データをトリップごとに分離し，個々のトリップごとの算出（移動量，移動時間，平均位置，平均時間，直線移動からのずれ，等速移動からのずれなどの算出が考えられる）を行う．この場合は，扱う対象のデータに，センサーデータに，トリップデータが加わる．なお，本稿では，センサーデータからのトリップデータへの推定法は対象外とする．なお，トリップデータを用いることでクレンジング対象となるレコードを発見できる可能性があり，トリップデータとセンサーデータは一体のシステムで扱うのが便利である．

３．Amazon Web サービスの概説

Amazon Web サービスは，コンピュータ，ストレージ，データベースなどの種々のインフラや，人工知能，IoT 等の種々のサービスを提供する商用クラウドサービスである．種々の既存のクラウドサービスの比較や評価を行うは本稿の対象外である．Amazon Web サービスを一例として，クラウドサービスでのセンサーデータの取り扱いについて議論したい．

クラウドサービスは，インターネットを経由して，種々のサービスを借りて利用するものである．サービスのための機器類の設定，設置等を気にせず，すぐに使い始めることができる．2.1 で説明したマップの処理や，SQL の処理や，多数のファイルに対する同一処理などを並列処理できる機能がある．

以下，Amazon Web サービスについて，その主要要素である EC2, S3 (simple storage service), File Gateway, lambda, Redshift を概説する．

・EC2

EC2 は，「コンピューティング」の能力を提供するクラウドサービスである．さまざまな種類，性能のものを選択可能である．EC2 の上に，種々のオペレーティングシステムをインストールして使うことができる．

・S3 (simple storage service)

S3 は，「ストレージ」の能力を提供するクラウドサービスである．ファイルは，S3 内のオブジェクトとして格納され，アップロードやダウンロードができる．S3 はファイルシステムではないので，ファイル操作（fgets, fputs など）を行うことはできない．容量は超大容量であり，従量課金である．

・File gateway

File Gateway は，S3 内のファイルのオブジェクトを，NFS マウント可能なファイルシステムとして取り扱えるようにするためのサービスである．ファイルシステムとして扱えるので，ファイル操作（fgets, fputs など）を行うことができる．Linux の ls コマンドのファイル操作コマンドや，一般のアプリケーションでのファイル操作もできる．仕組みとしては，ファイルアクセスによって，S3 内のファイルのオブジェクトが，File gateway システム内にダウンロード，キャッシュされるとともに，ファイルの更新により，S3 内にファイルのオブジェクトアップロードされるものである．

・lambda

lambda は，ファイル操作などを並列で実行できる能力を提供するサービスである．例えば，ファイルの

変換（1つのファイルから別のファイルを作る操作）を行うプログラムを Python で書き，lambda に登録しておくと，多数のファイルに対するファイルの変換処理を並列実行（このとき，ファイルの変換変換を行うプログラムが多数同時に動く）できる機能がある．

・Redshift

Redshift は SQL が動くデータウエアハウスシステムのサービスである．Copy コマンドにより，外部からの CSV ファイルの copy ができる機能などがある．二次索引の機能はない．Redshift のテーブルは，複数のノードに分散配置され，並列処理される．ノードへの分散配置のスタイルは，全分散，均等分散（ラウンドロビン方式），キー分散（ある属性の値による分散．属性値が同じものは同じノードに配置される）の種類を選ぶことができる．分散配置のスタイルを変えても実行結果は変わらないが，SQL などの処理にかかる時間が変わる（特に複数テーブルの結合を行う場合には，注意深く分散配置のスタイルを選ぶ必要がある）．

4．センサーデータ管理システムの設計

3. で説明した通り，クラウドサービスは，次のような特色がある．これら特色を考慮しながら，センサーデータ管理システムの設計を行う．

- 超大容量で，容量拡張が容易なストレージ
- 標準的な性能で，スポット借り上げが可能であり，即時にオペレーティングシステムのセットアップができるコンピュータ
- 多数ファイルに対する同一ファイル処理を並列で行える機能
- 高い並列度の SQL 並列処理ができる機能（SQL コンパイラと実行環境）

4．1　システム設計内容

本稿では，センサーデータは，リアルタイムで，クラウドシステムにアップロードすることは考えない．いったん，中継となるコンピュータで蓄積後，1日程度の頻度で定期的に S3 にファイルとしてアップロードする．

S3 にアップロードされたセンサーデータは，クラウドシステム内のデータウエアハウスである Redshift に copy する．そして，クレンジングの処理が行われる．2.1 で説明したように，クレンジング関数のプログラムを実行し，センサーデータのレコードごとに，クレンジング関数の数だけの true, false の値を判定し，その結果を記録する．この記録は，クレンジングの検証に用いる．以上の処理において，センサーデータのレコードの更新や削除は行いわない．以上の copy およびクレンジングの処理は，アップロードの頻度に合わせて，1日程度の頻度で定期的に行われるとする．クレンジングの処理は，次の (1) あるいは (2) で行う．そのどちらかで行うかは，個々のクレンジング関数の特性によるとする．クレンジング関数の特性は 4.2 で説明する．

(1) lambda による処理
Python プログラムを lambda 上で実行する．このときは，S3 の処理で行う．並列処理の単位は個々のファイルである．この方式は，複数種類のセンサーデータの組み合わせでクレンジングを行う場合には使わない．SQL の知識がなくても，値の比較，文字列の操作の程度でプログラム可能である．

(2) Redshift のよる処理
SQL と，SQL のユーザ定義関数の機能により組み込んだ Python の関数を Redshift 上で実行する．並列処理の単位は，Redshift ノードに分散配置されたテーブルである．この方式は，複数種類のセンサーデータの組み合わせでクレンジングを行う場合でも使える．SQL を使うため，プログラムの維持，改変は容易である．Redshift の設定により，性能が左右される．

クレンジングの結果，正当と判断されたセンサーデータのレコードは，次のように扱う．

- Redshift に格納する．このことで，データのグループ化および集計集約処理（2.3 で説明）を Redshift で行うことを可能にする．
- あわせて，File gateway を経由して，クラウドシステム外にファイルとして，ダウンロードできるようにする．これによって，クラウドシステム外のリレーショナルデータベース管理システム及び Redshift でも処理が行えるようにする．特に，二次索引を行うような処理については，クラウドシステム外のリレーショナルデータベース管理システムでの処理を想定する．

グループ化および集計集約処理では，レコードの選択（2.3 で説明）が行われる．Redshift への格納においては，テーブルの分散格納のスタイルについての設定ならびにソートキー（Sort Key）の設定が必要である．ソートキーは，SQL の group by, order by の性能向上に影響するものである．Group by, order by にどの属性を使用するかを前もって決定することが難しい場合には，それでも，性能面で有利となるように，分散格納のスタイルは全コピーとし，ソートキーは，グループ化および集計集約処理の担当者が必要に応

じて動的に生成するものとする. ソートキーが範囲選択に有効かどうか, 将来実験的に確認し, 有効性が確認できた時点で, 改めて, ソートキーの作成方針について再検討することとしたい.

Redshift の処理では, ソートキーが使われない限り, 全データのフルスキャンが行われる. フルスキャンは, データサイズに比例して処理時間がかかる. そのために, センサーデータが含むセンサーデータの計測日時の値と, その他属性値によるテーブル分割を行う. すなわち, センサーデータは, その種類ごとに, 一般利用者からは, 「様式１」, 「様式２」のような分かりやすいテーブル名で扱えるようにするが, 実際には, 複数のテーブルに分割し, さらには, それぞれのテーブルが, 複数の Redshift ノードに分散配置される. そのために, 「様式１値０から１００日付２０２１年１月」のように, 属性値の範囲と日付の範囲によりテーブルに分割するとともに, ビューの機能により, 「様式１」, 「様式１日付２０２１年１月」のようなテーブル名でも扱えるようにする.

出発から到着までのトリップ（origin destination trip）については, 次の 2 種のデータを, Redshift に格納するとともに, ファイルとしてクラウドシステム外にダウンロードできるようにする.

(1) １つのトリップを１レコードとするトリップデータ
(2) トリップとセンサーデータのレコード間の１対多関係に関するデータ

なお, センサーは, 異種が混在することがある. さらには, センサーの改善等による属性の追加や属性の定義（コードの意味, 値域, 桁数など）の変更がありえる. これらのことは, 本稿の対象外とする.

４．２　センサーデータのクレンジングの特性

クレンジングには, 次に示すように, さまざまな種類があり, システム設計上の注意を要する.

<u>タイプ①　個々のレコード内で判定できるもの</u>

(1) 異常値
属性値のとりえる範囲に制約があるもの. 制約に反する場合は異常値である.

(2) 本来 NULL 値をとりえない属性が NULL となっているもの
この場合, NULL になっている場合は異常値である.

(3) センサー側での妥当でない処理
センサー側で集計処理, 判定処理などが行われ, その判定結果がレコードの中に含まれている場合で, その値が, 同一レコード内の他の属性値とつじつまが合わない場合.

(4) センサー側でのクレンジングによる除外判定
センサー側でクレンジング処理が行われ正当でないデータ（除外すべきデータ）であると判定されている場合

<u>タイプ②　近接データの追跡を必要とするもの</u>

(5) 隣接計測における異常
移動体の計測などで, 隣接する 2 回の計測において, 変化量に制約があるもの. 制約に反する場合は異常値である.

(6) センサー稼働開始直後における異常
稼働開始直後に, 計測が安定しない（例えば, GPS の即位が安定しない）などの理由で, 誤差の大きな計測値になっているもの.

(7) センサー側での補正が妥当でないもの（GPS の補正遅れなど）
センサー側での妥当でない補正が始まり, 複数の観測点について, 妥当でない補正がしばらく続くもの. これは, 上記 (3) では補正の開始点と終了点しか判別できないため, 継続しての追跡を必要とする.

<u>タイプ③　同一種の全センサーデータのスキャンによって判定するもの</u>

(8) 一意制約
全観測にわたって, 同じ値が 2 回以上でないことが分かっている属性値. 2 回以上出た場合には, 計測の重複などの異常が考えられる.

以上の 3 つのタイプのうち, ①は, 4.1 で説明した lambda による処理, Redshift による処理のどちらの処理であっても効率よく処理できる. 性能評価については, 処理対象のファイル数, Redshift の設定によるため, 今後の課題とする. 以上の 3 つのタイプのうち②および③は, 処理の特質から, Redshift による処理よりも lambda による処理の方が, 性能面で有利である.

正当でないデータについては, その要因分析が重要である. そのため, 個々のクレンジング関数について, クレンジングの結果, 正当でないと判断されたデータについては, それのみを格納した新たなファイルを S3 内に作るとともに, それを, Redshift に copy することで, 要因分析を可能とする.

４．３　データのバックアップ, ダウンロード

ここでのバックアップは, データの人為的なミスによる棄損に備えるものである. 以上の設計は, 全データは, 2.2 で説明したデータ蓄積の方式で, S3 内のファイルのオブジェクトとして格納される. そして必要な部分が Redshift にコピーされるという設計にしている. そのため, データのバックアップとしては S3 内の全データをバックアップしておくことで十分である. ダウンロードについても S3 内のファイルのオブジェクトを直接ダウンロードするか, あるいは, File Gateway を使ってダウンロードする. 直接ダウンロー

ドは，Python 等のプログラムから，小数のファイルをダウンロードするときに向く．File Gateway を経由したダウンロードは，ファイル転送用のソフトウエアを使い，ファイルの一覧を見ながら，sftp 等でダウンロードするときに向く．

４．４　運用コンピュータ

4.1，4.2，4.3 で説明してきたシステム操作は，運用期間中は定型化できないと想定している．そのために，システムをインタラクティブに操作できるための運用コンピュータが必要である．そのために，Amazon EC2 の Windows パソコンのインスタンスをダイナミックに生成できるように準備しておく．この上では，運用担当者は，lambda のコマンド，Redshift で動く SQL プログラム，ファイルのブラウズ（確認のため）やクラウドシステム内でのバックアップ操作を行う．いずれも特別な操作ではなく，シェルのコマンド等を実行して操作できるものである．このことは，捜査記録を残しておくことにも便利である．

種々の操作の進行状況の把握，データ全体の一貫性のチェックについても運用コンピュータを利用して行う予定である．その具体的な設計は，実データの様態に依存するため，今後の課題とする．

５．おわりに

クラウドサービスは，すぐにコンピュータ等を使い始めたい場合，簡単にストレージの容量拡張を行いたい場合に向く．今回の設計を通して，センサーデータのクレンジングの並列処理，センサーデータのグループ化の集計集約の並列処理を行うようなシステムが，容易に実現できる見通しが立った．これら処理は，常時一定の負荷というよりは，単発的に高い負荷がかかる見込みがあり，クラウドサービスの方が運用面で優れる可能性がある．

一方で，リレーショナルデータベース管理システムの SQL 処理は，並列処理ではなく，二次索引や，システム内の大容量データ向けのソートや結合のアルゴリズムを使っての高速処理を行うもので，高速処理できるための仕組みが違う．データウエアハウスとリレーショナルデータベースシステムとの性能比較は将来の重要な研究課題としたい．

謝辞

本稿の作成にあたり，センサーデータの取り扱いに関するさまざまなご示唆，最新技術動向，知見・知識を提供くださった ETC システム株式会社片山賢治氏に感謝します．