Wheren : Z 曲線を用いた時空間情報の検索の高速化

北原 壮[†] 中西泰人[††,†††]

近年，GPS 備えた携帯電話端末が普及するにつれ，位置情報を用いたサービスが増えつつある．こうしたサービスを提供するためには，緯度経度および時刻を持つ大量の時空間情報を保存するデータベースを高速に検索する必要がある．そのためには空間インデックスを用いることが一般的であるが，空間インデックスを利用できないデータベースを利用する場合には，通常のインデックスを用いながら空間的な情報を検索する手法が提案されている．そうした手法の中で緯度経度だけでなく時刻も含んだ情報を取り扱える手法は筆者らが知る限りでは存在しない．そこで本稿では，空間充填曲線の一つである Z 曲線を用いて緯度経度および時刻の情報を一次元に変換することで時空間情報を検索する手法である Wheren を提案すると共に，その評価を行う．

Wheren : a Speed up Technique for Searching Spatiotemporal Data using Z-Curve

Soh Kitahara[†] and Yasuto Nakanishi[††,†††]

In recent years, location-based services become common with the increasing amount of mobile phones with GPS. It is necessary to search a lot of spatiotemporal data in database speedly in these services. To do this, it is common to use spatial index. It is proposed that searching spatiotemporal data technique to use normal index for the database don't have it. It isn't exist to search not only latitude and longitude but also time in our knowledge. So in this paper, We propose and evaluate Wheren that is translating to one dimension from spatiotemporal data using Z-Curve that is one of space-filling curve.

1. はじめに

近年，GPS を備えた携帯電話が広く普及し，GPS から取得される位置情報をインターネット上の Web サービスに高い頻度で送信するアプリケーションが多く開発されるようになった．また複数のユーザから位置情報が送信されることを前提とした Web サービスも多く提供されている．そうしたサービスの例には，チェックインサービスの foursquare[☆]，すれ違いサービスの EncountMe[☆☆] などがある．

このような Web サービスでは利用者の位置情報だけでなく時刻情報を用いてサービスを提供する．例えばすれ違いサービスを提供するには，ある緯度経度および時刻の範囲の中にどのユーザ達が存在しているかを高速に検索出来る必要がある．

このような Web サービスでは大量のデータを扱うために，MySQL 等のリレーショナルデータベースが多く使われており，また大量のデータの中から高速に検索するためにインデックスを使用することが一般的である．インデックスを使うことでディスクにアクセスすることなくメモリから高速に検索することが出来る．インデックスには B 木や R 木を元にした手法を用いることが多く，それらを改良した様々な提案がが存在する[5]．特に位置情報を用いた Web サービスでは，ある緯度経度，時刻を起点としてその周囲を検索するために，K 近傍法[3] のような距離空間における検索を行える必要がある．そのために使われるインデックスは，空間インデックスと呼ばれる．

Web サービスに広く使われているリレーショナルデータベースである MySQL で用いられるストレージ

† 慶應義塾大学大学院政策・メディア研究科
Graduate School of Media and Governance, Keio University
†† 慶應義塾大学環境情報学部
Faculty of Environment and Information Studies, Keio University
††† 科学技術振興機構　さきがけ
JST PRESTO
☆ http://foursquare.com
☆☆ http://encountme.com/ja/

エンジンには MyISAM☆と InnoDB☆☆がある．一般的に MyISAM は参照系クエリに対しては高速であるが，更新を行う際はテーブルロックを行うため，更新系クエリがある場合は行ロックを行う InnoDB が使われており，この特長から InnoDB は MyISAM に比べて多く利用されている[4]．しかしながら，MyISAM が空間インデックスをサポートしているのに対して，InnoDB は B 木を元とした通常のインデックスだけしか利用できない．そのため，位置情報を用いたサービスを提供しようとした場合には InnoDB を選択することは難しい．このように，位置情報を用いた Web サービスを提供するにあたってあらゆる実装のデータベースで空間インデックスが利用できる訳ではないという問題点がある．

この問題を解決する手法として，空間充填曲線の一つである Z 曲線を用いて多次元情報を一次元情報へ変換することで空間的な検索を可能とする手法がある．この手法を用いてデータベースを高速に検索する手段には，大きく二つに分けることができる．一つは，Ramsak ら[2] のようにデータベースのインデックスとして実装する手法である．もう一つは，GeoHash☆☆☆のように緯度経度の情報を文字列に変換し，その文字列をデータベースへ記録して通常のインデックスのように利用する手法である．しかし，上述したように InnoDB のように前者の手法を使うことが出来ないデータベースでは，後者の手法をとる必要がある．

GeoHash は，Z 曲線を基にして，緯度と経度の二つの値を一つの文字列へと変換する．その文字列は，その緯度と経度を中心にした範囲を示すことが出来る．また，文字列に文字を追加すればその範囲に含まれるより小さな範囲を，また文字を削除すればその範囲を含むより大きな範囲を指し示すことが出来るという特長がある．また，通常のインデックスであれば完全一致検索または前方一致検索を高速に行うことが出来る．それと同じように，GeoHash を用いて検索を行う場合は，データベースへ検索する際に使う文字列の長さよりももある程度長い文字列をインデックスとして登録しておき，前方一致検索を行う．

しかし上述したような位置情報を用いた Web サービスでは緯度と経度だけではなく，時刻も扱う必要がある．そこで，緯度，経度，時刻の三つの情報を Z 曲線を利用して一つにまとめ，検索をすることが出来れば，空間インデックスを用いることができない場合に，GeoHash を使った検索よりもより高速に検索を行えると考える．また，GeoHash のように緯度経度を文字列で表す技術はジオコードと呼ばれており様々なものが提案されているが[1]，時刻情報を含む三次元情報を表す物は筆者の知る限りでは提案されていない．そこで，本研究では，緯度経度，時刻を合わせた三次元情報を，Z 曲線を利用して一次元情報に変換する手法である Wheren を提案する．

☆ http://dev.mysql.com/doc/refman/5.5/en/myisam-storage-engine.html
☆☆ http://www.innodb.com/
☆☆☆ http://geohash.org

2. Wheren

2.1 Z 曲線と GeoHash

Z 曲線は多次元のデータを扱うことも可能であるが，図 1 に二次元の場合の Z 曲線を示す．Z 曲線は空間をアルファベットの Z を書くように充填し，充填したそれぞれの空間に以下のように一意な値を与える．

対象とする空間の X 軸 Y 軸の値をそれぞれ 2 進数に変換し一桁ずつ交互に組み合わせる．例えば，図 1 の左図の右下の空間の値を求める場合，2 進数は 01，00 となり交互に組み合わせると 0010 となる．2 進数に変換する際の桁数は空間の大きさによって異なる．図 1 の右図は，左図の各空間を一段階再帰的に分割したものである．この分割された空間は，分割される前の値にそれぞれ異なる値を連結した値を与えることができる．例えば右下四つの空間はそれぞれ 001000，001010，001001，001011 という値を持つ．そのため，一つ大きな範囲の空間を検索するには，空間を表す値の後方二文字を比較せず，前方の四文字だけを用いればよい．

GeoHash では，Z 曲線を基に分割をする際の緯度と経度の個数を交互に変更する．まず，二次元空間を緯度を四つ経度を八つに分割し，それぞれの空間を 5 桁の 2 進数で表す．さらにその空間を分割する場合は緯度を八つ経度を四つの空間に分割する．そして求められた 5 桁の 2 進数それぞれを 32 進数へ変換し，数値から文字列に変換する．ゆえ，一文字が一つのレベルを表し，これを後方から削除することで，より大きな空間を示すことが出来る．

2.2 Wheren のアルゴリズム

Wheren ではこれに時刻情報を加える．経度には東経 180 度と西経 180 度，緯度には北緯 90 度と南緯 90 度という上限と下限ああるが，時刻にはあらゆるサービスに共通の上限と下限がないため，サービス毎に時刻の上限と下限を予め決めておく必要がある．時刻の表現方法と上限と下限の決め方は任意であるが，本稿では，時刻情報には UNIX 時刻を用いデフォルトの

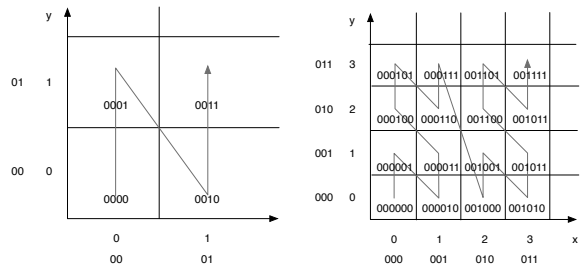

図 1 Z 曲線で空間を分割した場合の各空間の値

値を，下限を 1970 年 1 月 1 日 0 時 0 分 0 秒，32bit の int で表現できる秒数が下限から経過した 2038 年 1 月 19 日 3 時 14 分 7 秒とした．

Wheren は三次元空間を分割するが，その際に図 2 の左図のように緯度を四つ経度を四つ時刻を四つと右図のように緯度を四つ経度を四つ時刻を八つに分割する二種類を用意した．前者は一つの空間を 6 桁の 2 進数で表し 64 進数で文字列へと変換し，後者は 7 桁の 2 進数で表し 128 進数で文字列へ変換する．これにより後者は前者と比べてレベル間での時刻の範囲が大きく変化する．なお，今後本稿では前者を 6 ビット Wheren，後者を 7 ビット Wheren と呼ぶ．また，7 ビット Wheren は文字列へと変換する際に 128 進数を用いるが，英数字一文字ずつでは 128 種類を表すことが出来ないため，二文字で表すこととした．ゆえ，GeoHash や 6 ビット Wheren と異なり 7 ビット Wheren では文字列に対しレベルの変更を行う場合は二文字ずつ行う必要がある．

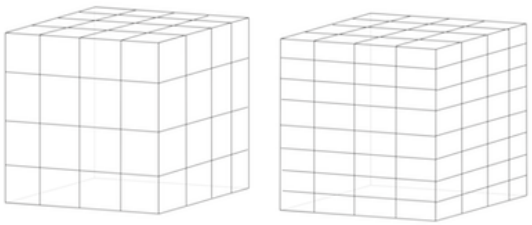

図 2 64 進数，128 進数を選んだ場合の空間の分割方法

GeoHash と同様に，Wheren も変換した文字列によってある緯度経度および時刻を中心とした三次元の範囲を示し，その文字列の長さによってその領域の大きさが変わる．本稿では文字列の長さをレベルと呼ぶ．レベルが大きいほどより狭い範囲を示し，レベルが小さい程広い範囲を示す．

2.3 Wheren の利用例

Wheren の実装として Perl モジュールを作成した☆．下記の例は 7 ビット Wheren を用い，時刻の上限下限の値をデフォルトのままで利用するものである．この中の encode メソッドの引数に，緯度経度および時刻，レベルを渡す．緯度に 35.3137 度，経度を 139.4863 度，時刻を 2011 年 1 月 1 日 0 時 0 分 0 秒の UNIX タイム，レベルを 7 を渡した場合，x0y240t342v1n3 という文字列が返ってくる．この文字列は，緯度が 35.310059 度から 35.321045 度，経度が 139.482422 度から 139.504395 度，時刻が 2010 年 12 月 31 日 23 時 59 分 10 秒から 2011 年 1 月 1 日 0 時 16 分 14 秒の範囲を表す．そして decode メソッドはこれらの値の中点を返す．

```
use Data::Wheren::7Bit;

my $wheren = Data::Wheren::7Bit->new;
my $hash = $wheren->encode(
  $lat, $lon, $epoch_time, $level
);
my ($lat, $lon, $epoch_time) = $wheren->decode(
  $hash
);
```

また，上述したように 7 ビットの場合は二文字で一つのレベルを表す為，得られた文字列の後ろ二文字を削除することでより大きい範囲を示す．先ほど得られた文字列から n3 を削除した x0y240t342v1 という文字列は，緯度が 35.288086 度から 35.332031 度，経度が 139.482422 度から 139.570312 度，時刻が 2010 年 12 月 31 日 21 時 59 分 42 秒から 2011 年 1 年 1 月 0 時 16 分 14 秒の範囲を示す．

3. 評　　価

緯度経度時刻の全てを範囲検索，GeoHash と時刻情報の範囲検索，Wheren のみを用いた検索の速度の比較を行った．

この評価では，データベースは MySQL 5.5.14 ，ストレージエンジンは InnoDB を利用した．テーブルのスキーマは，表 1 とする．Wheren に関するカラムは wheren_7bit_normal，wheren_7bit_half，wheren_6bit_normal，wheren_6bit_half の 4 種類を用意し，それぞれ 7 ビット Wheren を用い時刻の上限下限をデフォルトの値とその半分の値，6 ビット Wheren を用い時刻の上限下限をデフォルトの値とその半分の値を用いて変換した文字列を記録した．また，GeoHash，

☆ https://github.com/soh335/p5-Data-Wheren

Wheren 共にレベル 10 の文字列を登録した．

Field	Type	Length
id	INT	11
lat	DECIMAL	9,6
lon	DECIMAL	9,6
wheren_7bit_normal	VARCHAR	20
wheren_7bit_half	VARCHAR	20
wheren_6bit_normal	VARCHAR	20
wheren_6bit_half	VARCHAR	20
geohash	VARCHAR	20
created_at	DATETIME	

表 1　テーブルのスキーマ

検索の対象となるのデータとして 500 万件をランダムにデータベースへ記録した．ランダムに生成する際の最小の範囲を緯度を 35.50 度から 35.60 度，経度を 139.50 度から 139.60 度の共に 0.1 度の範囲とし，0.1 度ずつ 1.0 度まで増加させ，時刻は 2011 年 1 月 1 日 0 時 0 分 0 秒から 2011 年 1 月 5 日 0 分 0 月 0 秒の間とした．

これらのデータに対し検索範囲は，緯度経度の最小の範囲をそれぞれ 35.5475 度から 35.5525 度，139.5475 度から 139.5525 度の共に 0.005 度の範囲とし，0.005 度ずつ 0.05 度まで増加させ，時刻の最小の範囲を 2011 年 1 月 2 日 23 時 59 分 30 秒から 2011 年 1 月 3 日 0 時 0 分 30 秒の間の 1 分間，次の値を 2011 年 1 月 2 日 23 時 55 分 0 秒から 2011 年 1 月 3 日 0 時 5 分 0 秒の 10 分間としそれ以降は 10 分間ずつ 3 時間まで増加させた．

また，実行される全てのクエリはインデックスを用いるようにし，GeoHash ではレベル 5 から 7，Wheren ではレベル 6 から 8 を用いて指定された検索の範囲を覆う複数の文字列で前方一致検索を行った．この生成した緯度経度の範囲を青，検索する緯度経度の範囲を赤として図 3 に示した．

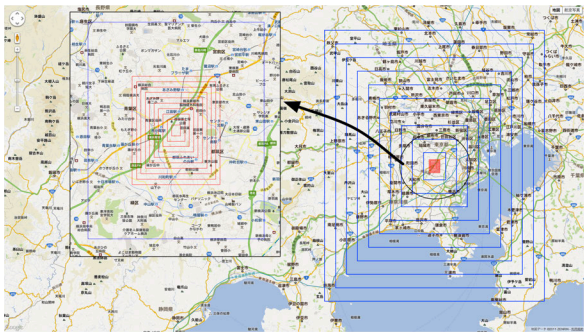

図 3　生成した緯度経度と検索する緯度経度の範囲

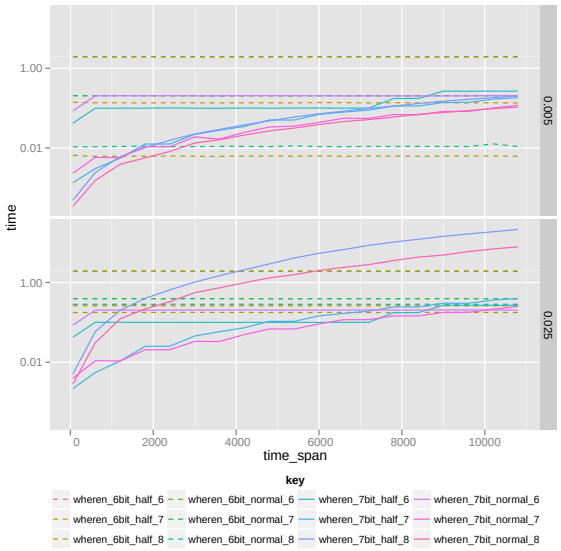

図 4　6 ビット Wheren と 7 ビット Wheren の特徴

図 4 の上のグラフは緯度経度の検索範囲を共に 0.005 度，下のグラフは共に 0.025 度とし，データの生成範囲を緯度経度共に 0.1 度とした場合に，6 ビット Wheren と 7 ビット Wheren の検索時間が時刻の検索範囲によってどう変化するかを示した．6 ビット Wheren は両グラフ共に時刻の検索範囲が大きくなっても検索速度に変化はないが，7 ビット Wheren は大きくなるにつれて検索速度が低下した．また，緯度経度の検索範囲が大きくなると，高いレベルで検索している場合ほど検索速度が低下することと，時刻の上限下限をデフォルトの値とその半分とした場合では検索速度に少しの違いが見られたが，検索する時刻の範囲による影響は大きな違いがないことが分かった．なお今後は，Wheren に関する比較項目が多いため特に特徴があった 6 ビット Wheren で時刻の上限下限の値をデフォルトの半分の値にしレベル 8 で検索した場合と，7 ビット Wheren で時刻の上限下限の値をデフォルトの値にしレベル 7 で検索した場合を扱うこととする．

図 5 の上のグラフは時刻の検索範囲を 600 秒，下のグラフは 6000 秒とし，緯度経度の検索範囲を共に 0.005 度とした場合に，データの生成範囲によって検索時間がどう変化するかを示した．時刻の検索範囲を 600 秒とし，GeoHash のレベル 5 で検索をした場合を除いて，データの生成範囲が小さくなるほど検索速度が低下した．

図 6 の上のグラフは時刻の検索範囲を 600 秒，下のグラフは 6000 秒とし，データの生成範囲を緯度経度共に 0.1 度とした場合に，緯度経度の検索範囲によっ

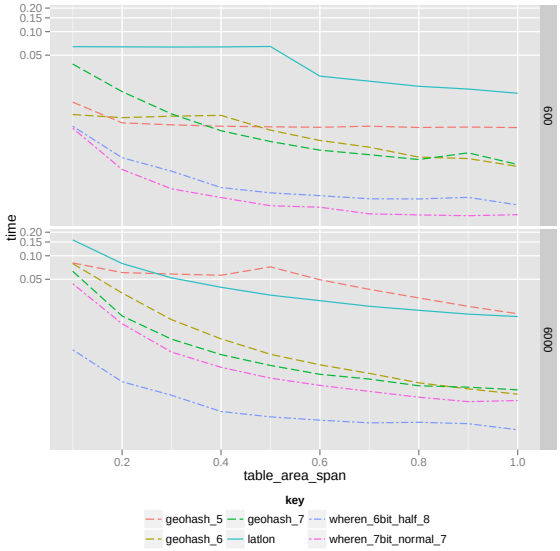

図 5　データの生成範囲による検索時間の変化

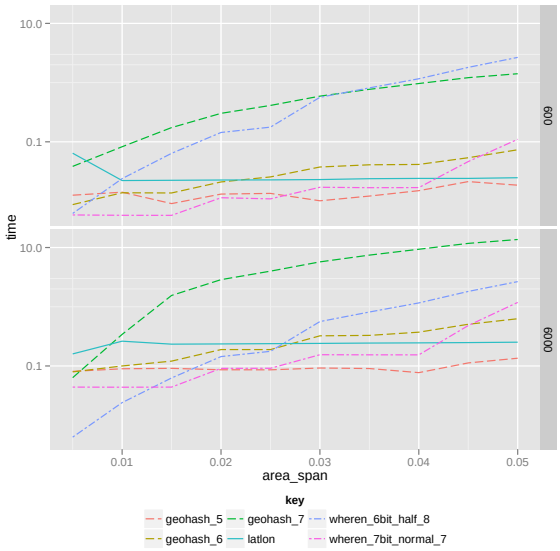

図 6　緯度経度の検索範囲による検索時間の変化

て検索時間がどう変化するかを示した．上述した通り，Wheren を用い高いレベルで検索をしている場合，緯度経度の検索範囲から大きく影響を受けていることが分かったが，GeoHash を用いた場合でもこの特徴が見られた．また，緯度経度時刻の全てを範囲検索で行った場合は緯度経度の検索範囲からの影響は少ないことが分かった．

4. 考　　察

GeoHash や Wheren を用いて高いレベルで大きい範囲を検索すると極端に検索速度が低下した．これは高いレベルの文字列はより小さい範囲を示すため，検索範囲を埋め尽くす為にはより多くの文字列が必要となることが原因と考えられる．

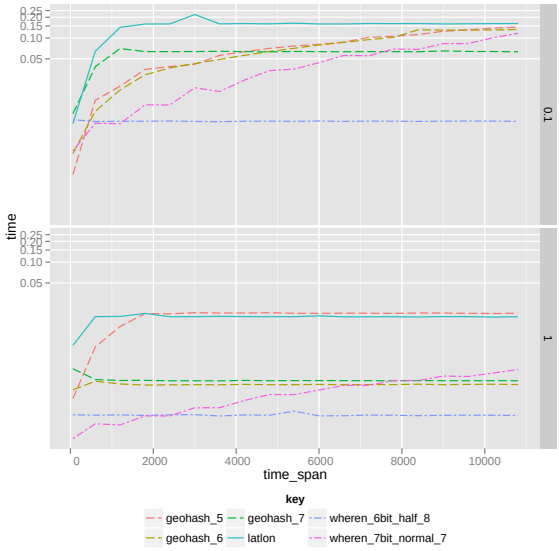

図 7　検索する緯度経度の範囲が共に 0.005 度の場合

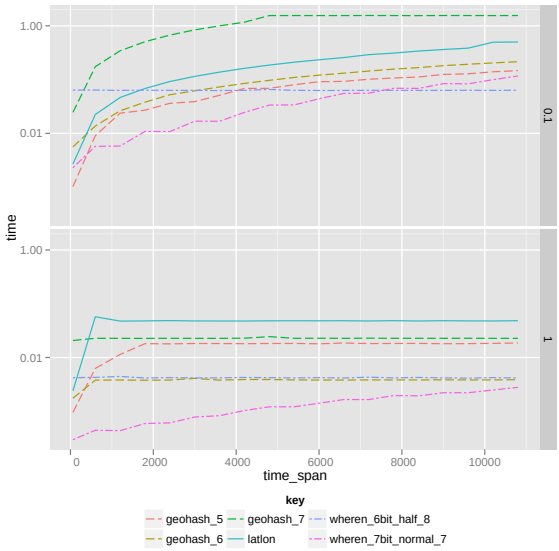

図 8　検索する緯度経度の範囲が共に 0.015 度の場合

図 7，8，9 の上のグラフはデータの生成範囲が緯度経度共に 0.1 度，下のグラフは 1.0 度とし，検索する緯度経度の範囲が順に共に 0.005 度，0.015 度，0.05 度の場合に，検索する時刻の範囲によって検索時間がどう変化するかを示した．図 7 からは，検索する緯度経度の範囲が共に 0.005 度の場合では 6 ビット Wheren でレベルを 8，時刻情報の上限下限の値をデフォルトの半分として用いると検索速度がもっとも速いことが

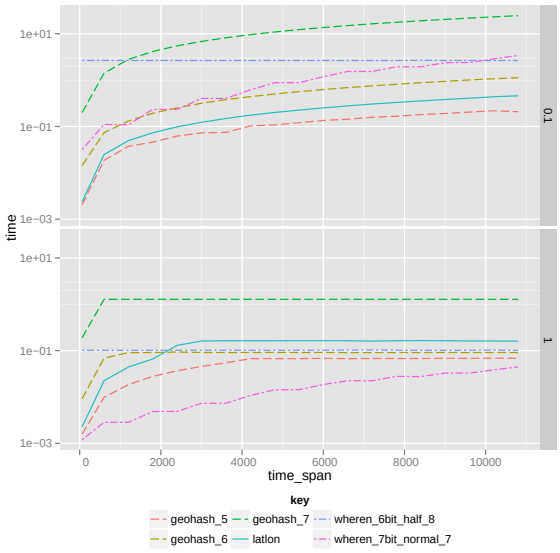

図 9 検索する緯度経度の範囲が共に 0.05 度の場合

分かった．図 8 からは，検索する緯度経度の範囲が共に 0.015 度の場合では 7 ビット Wheren でレベルを 7，時刻情報の上限下限の値をデフォルトのままとして用いる検索速度がもっとも速いことが分かった．図 9 からは，データの生成範囲が緯度経度共に 1.0 度の場合では 7 ビット Wheren でレベルを 7 として用いると他の手法より検索速度が速いが，範囲が共に 0.1 度の場合では GeoHash をレベルを 5 として用いた場合が最も早く，Wheren を用いた場合は検索速度が遅くなることが分かった．図 7, 8 においてもデータの生成範囲が狭くなると，Wheren と他の手法の検索速度の差が小さくなっている．データの生成範囲が狭くなるとデータの密集度が上がるため，Wheren が表す一つ一つの空間に存在するデータの量が多くなり，それが検索速度に影響を与えていると考えられる．

今回の評価では，ある範囲の中に 500 万件のデータを登録したが，地球全体を範囲とした場合，データの生成範囲が緯度経度共に 1.0 度の密集度では 3240 億件，0.1 度の密集度では 3 兆 2400 億件のデータを取り扱うことになる．今回の評価では，こうした大規模なデータを扱うための環境を用意することが出来なかったが，これほど大規模なデータを扱う場合には，複数の計算機を用いてデータベースで運用することが現実的である．密集度が一台の計算機辺り 1.0 度の範囲程度であれば Wheren を用いることで高速に検索出来る可能性があるが，扱うデータが極端に大きくなった場合，データベースの性能によっては本稿と同じような結果を得られないことも考えられる．より大規模なデータを分散して取り扱う場合の評価を行うことは，今後の課題の一つである．

また Wheren の問題点として，正確な検索を行うことには向いてないこと，出力されるログが読みづらいことがある．前者については，上述した通り正確な検索を行うには高いレベルを用いるために長い文字列を使って検索をする必要があるため，検索速度の低下を招いてしまう．そのため，検索速度よりも正確さが求められる場合には Whren は適していないと考えられる．後者については，今回の評価では，指定された範囲を埋め尽くす空間を求めてそれらを OR 句で繋いで検索をしたため，一つの SQL 文が長くなってしまうという短所があった．しかし，実際のサービスでデータベースを設計する際に，検索する範囲や適切な Wheren のレベルや時刻の上限下限の値について考慮することで無駄のない SQL 文を生成することが出来ると考える．

5. ま と め

本稿では，緯度経度および時刻を合わせた三次元情報を Z 曲線を利用して文字列に変換することで，その検索を高速化する手法である Wheren を提案し，その評価と考察を行った．データの生成範囲が緯度経度共に 1.0 度とし，緯度経度の検索範囲が共に 0.005 度の場合は最大 4.5 倍，0.015 度の場合は 7.5 倍，生成範囲が緯度経度共に 0.1 度とし，緯度経度の検索範囲が共に 0.005 度の場合は最大 2.6 倍，0.015 度の場合は 3 倍ほど高速であることが分かった．今後は Wheren が持つ特徴に適した Web サービスとそれに適切なレベルや時刻の上限下限の値について検討を行う．

参 考 文 献

1) 上田 直生, Venkatesh Raghavan, 地理的座標を表現するコードシステムと，目的に応じたコードの評価および選択手法の研究, 大阪市立大学学術情報総合センター紀要, Vol 9, pp. 29-35, 201003
2) Ramsak Frank et al., Integrating the UB-Tree into a Database System Kernel, Proceedings of the 26th International Conference on Very Large Data Bases, 2000
3) Belur V. Dasarathy, Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques, IEEE Computer Society Press, 1991
4) Bitner Jerad and Haug Nate, Large-scale web site infrastructure and Drupal, Linux J., 2010
5) Thomasian Alexander, Storage research in industry and universities, SIGARCH Comput. Archit. News, Vol. 38, pp. 1-48, 2010