ORACLE

MySQLで処理するGIS

～地球が丸いことを覚えたMySQL～

大塚 恒平/OTSUKA, Kohei
MySQL Community Team
日本オラクル株式会社
Principal Solution Engineer
MySQL Global Business Unit
2023年8月26日

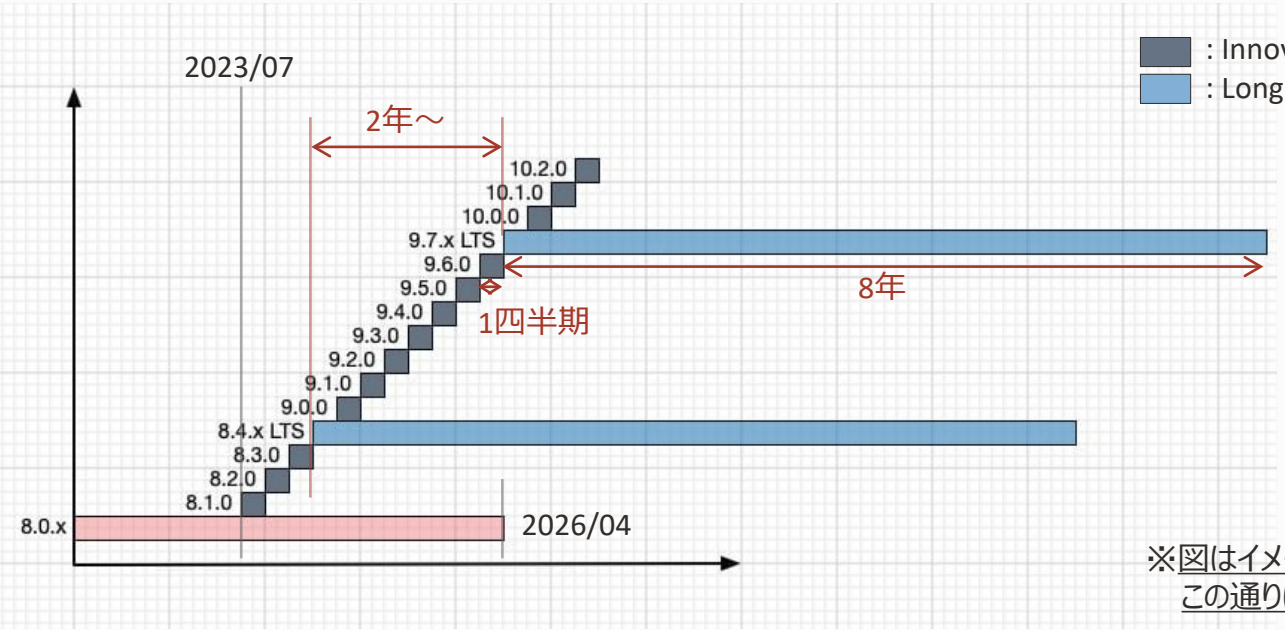
自己紹介

- 名前：大塚 恒平　（おおつか こうへい）
- 所属：日本オラクル株式会社
  MySQL Global Business Unit
- 役割：MySQLのプリセールス、MySQLの普及促進活動、など
- 専門分野：GIS、地図、地理などの業界で20年
- Github：kochizufan
- 出身：姫路
- 趣味：オープンソース開発、地方史研究（群馬、奈良など）、石造文化財研究

The world's most popular opensource database

世界で最も普及しているオープンソース データベース

MySQLの新バージョンポリシー：LTSとInnovation Releases



※図はイメージで、最終的なバージョン番号がこの通りになるとは限りません

MySQL Long-Term Support (LTS)

- 安定版: バグ修正とセキュリティパッチのみ
- 後方互換性
- 2年ごと
- サポートライフサイクル: Premier Support 5 年 + Extended Support 3年

MySQL Innovation Releases

- 最先端のイノベーション
- 容易にLTSとの移行可能
- 四半期ごと
- サポートライフサイクル: 短期

アジェンダ

1. GISとはなにか
2. MySQLで使える空間データ型
3. 空間データの表現方法
4. 座標系とSRID
5. 空間演算関数
6. 空間検索と空間インデックス
7. MySQLの利用方法

Appendix

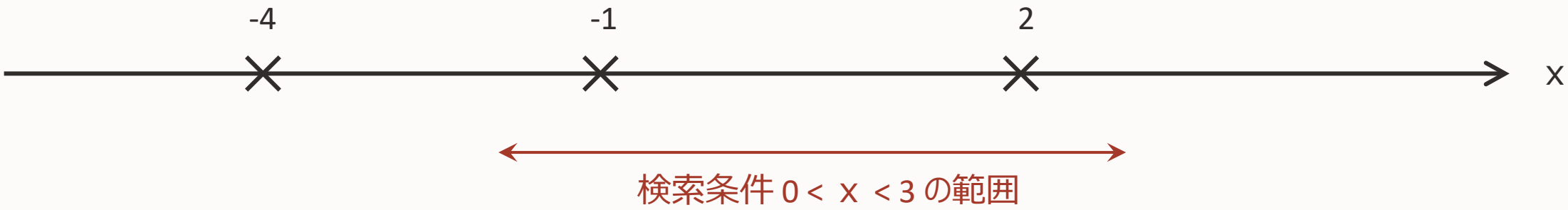
1. GISとはなにか

GISとはなにか

- GIS = Geographic Information System（地理情報システム）
- (x, y)座標、(経度, 緯度)座標など、2次元（以上）で表すデータを扱うシステム
- MySQL内ではspatial（空間の）というキーワードで扱われ、spatial用の型と関数が用意されている

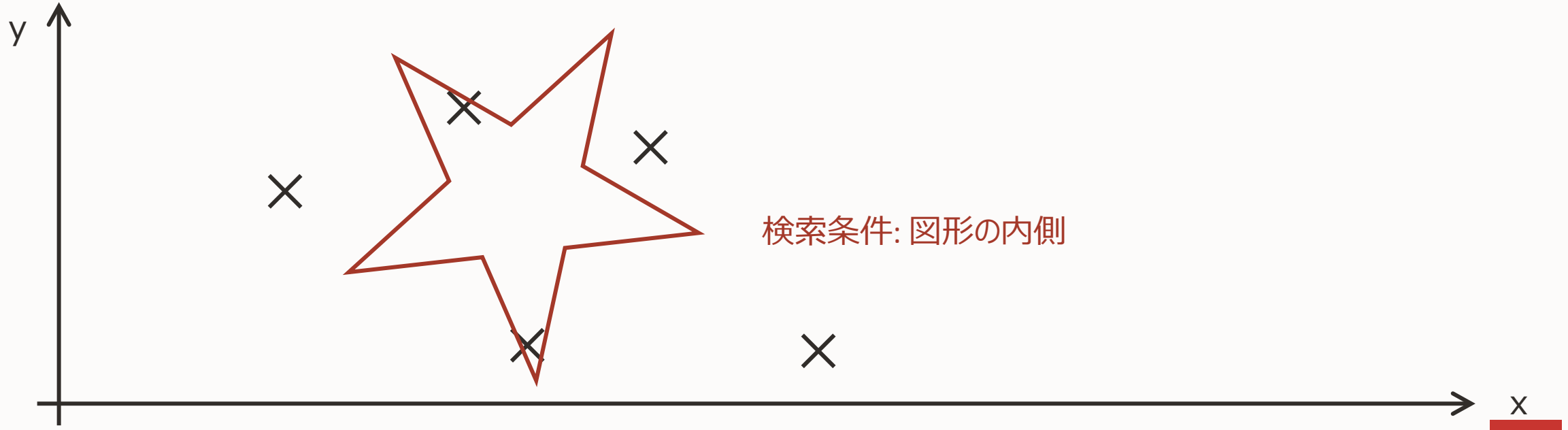
なぜGIS機能が必要か (1)

- 通常の一次元データ型: 数値型、文字列型（ロケールごとのアルファベット順）
  - 直線上にデータが存在
  - 昇順/降順など順番に並べれば高速に検索可能（インデックス）
  - 通常、検索条件も簡単（ある値より大きい/小さい、2つの値の間）



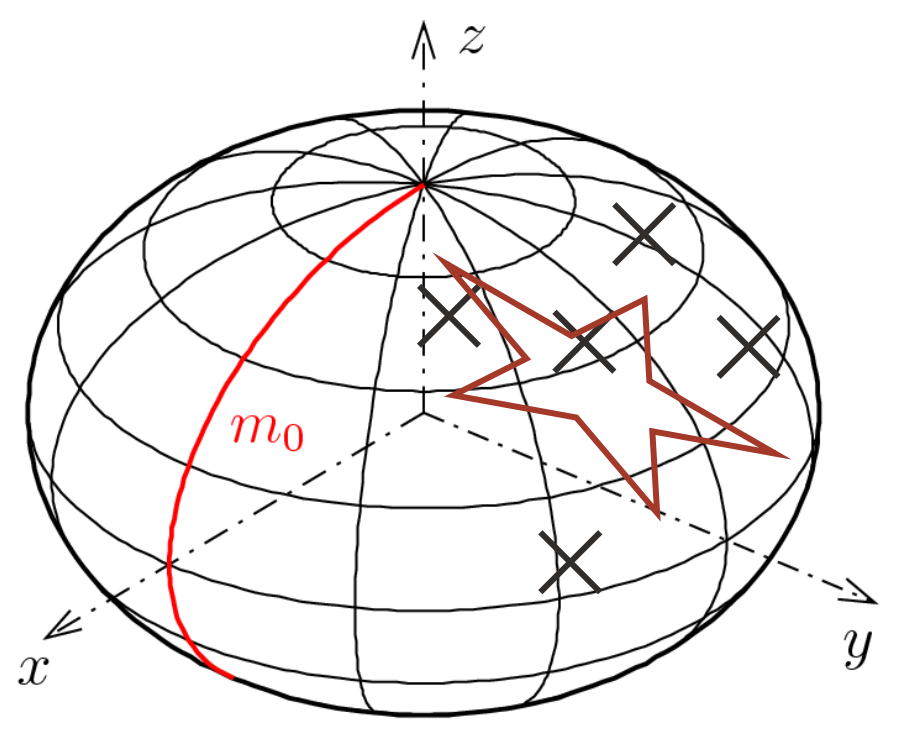
なぜGIS機能が必要か (2)

- 二次元以上のデータ:
  - 平面（空間）上にデータが存在
  - 2座標を個別カラムとして複合インデックスを張っても非効率
  - 検索条件も複雑 => 特別な型が必要



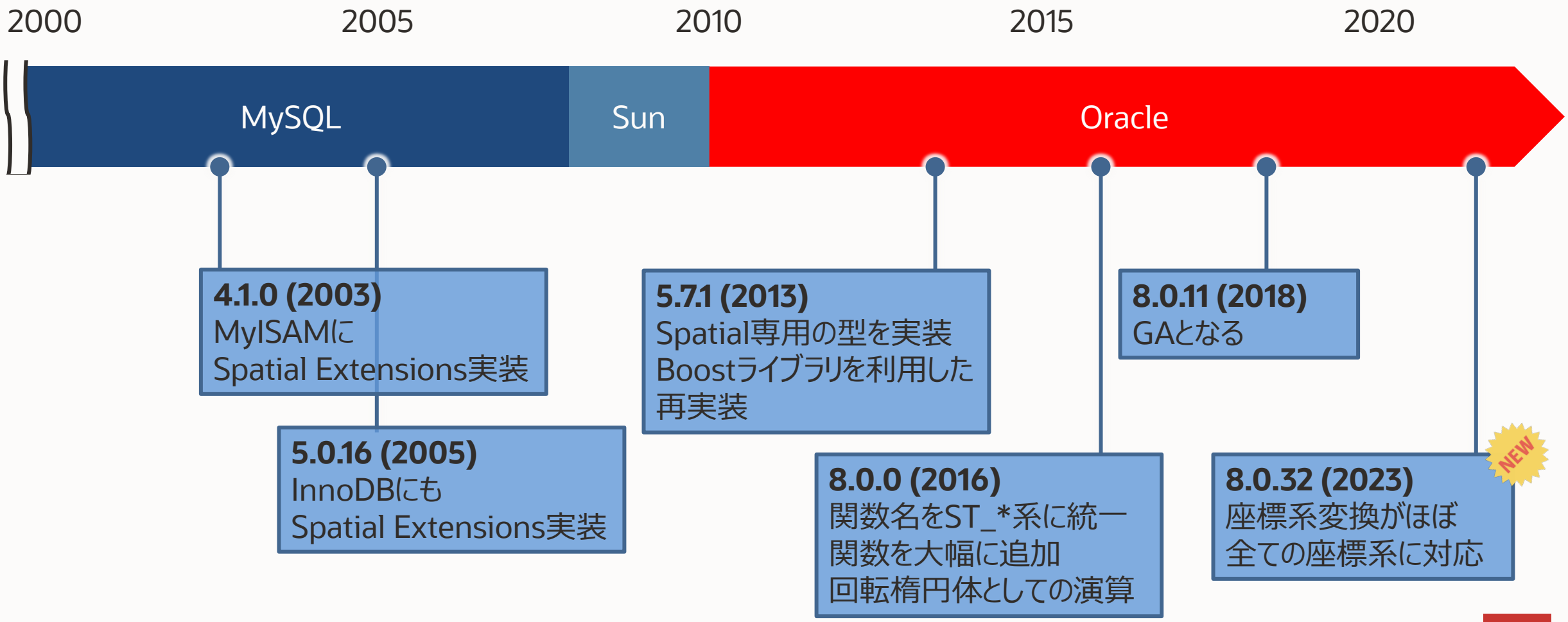
なぜGIS機能が必要か (3)

- 実際の地球は回転楕円体に近似できる:
  - 経度、緯度は極座標系 => 平面よりも問題が複雑に
  - さらにインデックス、検索条件も複雑に => 特別な型が必要



Wikimedia Commons, CC BY-SA 4.0

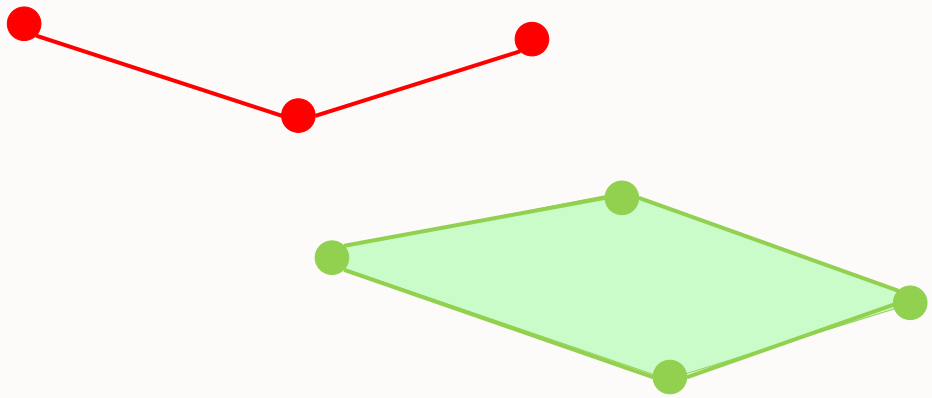
MySQLのGISの歴史



2. MySQLで使える空間データ型

空間データの型：単独オブジェクト

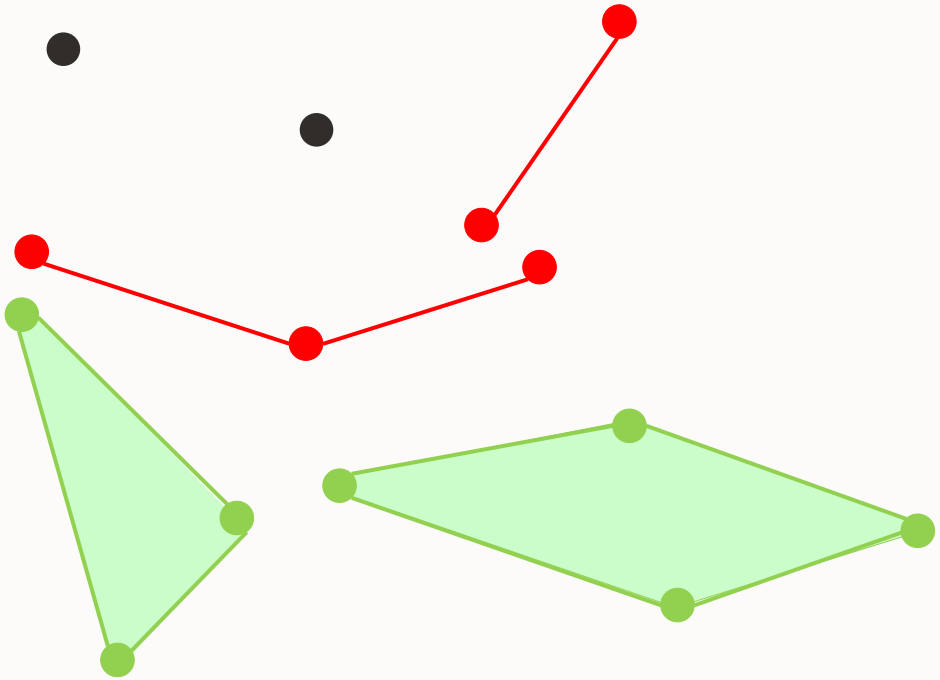
- POINT（点）
  - 例：店の場所
- LINESTRING（線）
  - 例：ルート（道筋）
- POLYGON（多角形）
  - 例：建物の外形



- GEOMETRY（POINT、LINESTRING、POLYGONを包括して扱える）

空間データの型：集合オブジェクト

- MULTIPOINT（複数の点）■
  - 例：複数の拠点からなる店の場所
- MULTILINESTRING（複数の線）■
  - 例：日毎のルート（道筋）
- MULTIPOLYGON（複数の多角形）■
  - 例：複数の建物からなる施設の外形



- それぞれの集合を扱える、MULTIPOINT、MULTILINESTRING、MULTIPOLYGON、GEOMETRYCOLLECTIONも存在

3. 空間データの表現方法

空間データの表現方法

- WKT（Well-Known Text）
  - 幾何学オブジェクトをテキストで表現するための仕様
- WKB（Well-Known Binary）
  - 幾何学オブジェクトをバイナリで表現するための仕様
- MySQLの内部表現
  - WKBの先頭にSRID（座標系を見分けるID）を追加したもの

- 他にも、Webでよく使われるGeoJSONなどのデータフォーマットもあり、
  様々なフォーマットからMySQLの内部表現に直接相互変換する関数もある

WKTの例

- POINT(15 20)　※区切りはスペース
- LINESTRING(0 0, 10 10, 20 25, 50 60)
- POLYGON((0 0, 10 0, 10 10, 0 10, 0 0))　※最初の点に戻る
- POLYGON((0 0, 10 0, 10 10, 0 10, 0 0), (5 5, 7 5, 7 7, 5 7, 5 5))

外側の境界線　内側の境界線

※ドーナツ状に中をくりぬくことも可能

WKTから空間データを生成する関数

- ST_GeomFromText(“WKT” [, SRID])
- ST_PointFromText(“WKT” [, SRID])
- ST_LineStringFromText(“WKT” [, SRID])
- ST_PolygonFromText(“WKT” [, SRID])
- ST_MultiPointFromText(“WKT” [, SRID])
- ST_MultiLineStringFromText(“WKT” [, SRID])
- ST_MultiPolygonFromText(“WKT” [, SRID])
- ST_GeometryCollectionFromText(“WKT” [, SRID])

戻り値はMySQLの内部表現（バイナリ）

WKBから空間データを生成する関数

- ST_GeomFromWKB(“WKB” [, SRID])
- ST_PointFromWKB(“WKB” [, SRID])
- ST_LineStringFromWKB(“WKB” [, SRID])
- ST_PolygonFromWKB(“WKB” [, SRID])
- ST_MultiPointFromWKB(“WKB” [, SRID])
- ST_MultiLineStringFromWKB(“WKB” [, SRID])
- ST_MultiPolygonFromWKB(“WKB” [, SRID])
- ST_GeometryCollectionFromWKB(“WKB” [, SRID])

戻り値はMySQLの内部表現（バイナリ）

内部表現からWKTやWKBに変換する関数など

- ST_AsText(内部表現)
- ST_AsBinary(内部表現)

- その他にも
  - ST_GeomFromGeoJSON
  - ST_AsGeoJSON

  など

ST_GeomFromText、ST_AsTextの動作例

```
MySQL  localhost:33060+ ssl  geo_test  SQL > SELECT ST_GeomFromText('LineString(1 1,2 2,3 3)');
+-------------------------------------------------------------+
| ST_GeomFromText('LineString(1 1,2 2,3 3)')                  |
+-------------------------------------------------------------+
|                  ??      ??      @       @       @       @ |
+-------------------------------------------------------------+
1 row in set (0.0015 sec)
MySQL  localhost:33060+ ssl  geo_test  SQL > SELECT ST_AsText(ST_GeomFromText('LineString(1 1,2 2,3 3)'));
+-------------------------------------------------------+
| ST_AsText(ST_GeomFromText('LineString(1 1,2 2,3 3)')) |
+-------------------------------------------------------+
| LINESTRING(1 1,2 2,3 3)                               |
+-------------------------------------------------------+
1 row in set (0.0025 sec)
MySQL  localhost:33060+ ssl  geo_test  SQL >
```

WKTから内部バイナリ変換

バイナリ

WKT => 内部バイナリ変換 => WKT

空間データを含むテーブル作成とデータの挿入/検索

```
MySQL localhost:33060+ ssl geo_test SQL > CREATE TABLE test(
                                       ->   id INT AUTO_INCREMENT PRIMARY KEY,
                                       ->   geom GEOMETRY
                                       -> );
Query OK, 0 rows affected (0.0272 sec)
MySQL localhost:33060+ ssl geo_test SQL > INSERT INTO test(geom) VALUES(ST_GeomFromText('POINT(1 1)'));
Query OK, 1 row affected (0.0086 sec)
MySQL localhost:33060+ ssl geo_test SQL > SELECT id, ST_AsText(geom) FROM test;
+----+-----------------+
| id | ST_AsText(geom) |
+----+-----------------+
|  1 | POINT(1 1)      |
+----+-----------------+
1 row in set (0.0010 sec)
MySQL localhost:33060+ ssl geo_test SQL >
```

空間データ型カラム

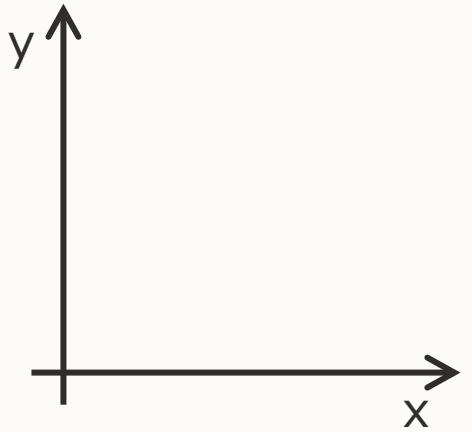
WKTから内部バイナリ

内部バイナリをWKT化して表示

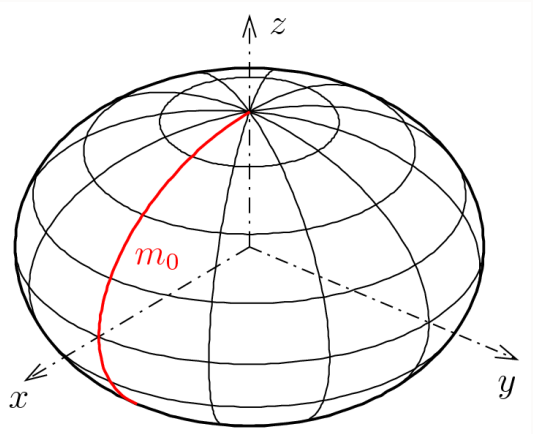
4. 座標系とSRID

地理座標系と投影座標系 (1)

- 地図としての座標系（平面）と、地球上の座標系（回転楕円体面）
- 平面：投影座標系（地図座標系）
  - xy座標系、平面の幾何学が成立する
  - 現実の面積、距離、方位、形などを全部正確に
    表現することは不可能（用途によって図法を使い分け）



- 回転楕円体面：地理座標系
  - 極座標系（経緯度）、曲面の幾何学
  - 国や地域、時代などによる歴史的経緯により
    複数の定義（近似法）が存在、一意ではない



Wikimedia Commons, CC BY-SA 4.0

地理座標系と投影座標系 (2)

- 空間データ型を持つ一般的なデータベースでの扱い方
  - 投影座標系型を処理する型： GEOMETRY型
    - 平面上での幾何計算を行う（地図としての世界を処理する）
      - 地図の性質次第で、現実世界での計算結果とは大きく異なる結果（長さ、面積、内外判定、角度、方向 etc.）
      - 計算が容易
  - 地理座標系を処理する型： GEOGRAPHY型
    - 楕円体面上での幾何計算を行う（丸い地球を処理する）
      - （楕円体近似の範囲内で）基本的には現実の計算結果と一致
      - 計算が複雑 => GEOGRAPHY型には対応していない関数もある

地理座標系と投影座標系 (3)

- MySQLでの扱い方
  - 空間データの型はGEOMETRY型しか存在しない

地理座標系と投影座標系 (3)

- MySQLでの扱い方
  - 空間データの型はGEOMETRY型しか存在しない
    - 地理座標系が扱えないわけではない => 誤解されがち
  - 座標系を自動判別し、処理を切り替え => 8.0以降、MySQLは地球が丸いことを覚えた 
    - 投影座標系では平面計算、地理座標系では楕円体面計算を行う
    - 大半の関数が両座標系に対応 ※
- 座標系を見分けるためのIDがSRID（Spatial Reference Identifier）
  - 指定しなければSRID=0、平面座標扱い（他座標系には変換できない）
  - 対応しているSRIDはINFORMATION_SCHEMA.ST_SPATIAL_REFERENCE_SYSTEMSで確認できる
  - 座標間の変換は、ST_Transform関数で行う => 8.0.32以降、大半の座標系に対応 

※ 地理座標系に未対応な関数はST_Centroid、ST_MakeEnvelope、ST_IsClosed、ST_Buffer（POINTには対応）のみ

対応している座標系の確認

```
MySQL  localhost:33060+ ssl  SQL > SELECT * FROM INFORMATION_SCHEMA.ST_SPATIAL_REFERENCE_SYSTEMS¥G
  ... 中略 ...
*************************** 396. row ***************************
                SRS_NAME: JGD2000 / Japan Plane Rectangular CS VI
                  SRS_ID: 2448
            ORGANIZATION: EPSG
ORGANIZATION_COORDSYS_ID: 2448
              DEFINITION: PROJCS["JGD2000 / Japan Plane Rectangular CS VI",GEOGCS["JGD2000",DATUM["Japanese Geode
tic Datum 2000",SPHEROID["GRS 1980",6378137,298.257222101,AUTHORITY["EPSG","7019"]],TOWGS84[0,0,0,0,0,0,0],AUTHOR
ITY["EPSG","6612"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.017453292519943278,AUTHORITY["
EPSG","9122"]],AXIS["Lat",NORTH],AXIS["Lon",EAST],AUTHORITY["EPSG","4612"]],PROJECTION["Transverse Mercator",AUTH
ORITY["EPSG","9807"]],PARAMETER["Latitude of natural origin",36,AUTHORITY["EPSG","8801"]],PARAMETER["Longitude of
natural origin",136,AUTHORITY["EPSG","8802"]],PARAMETER["Scale factor at natural origin",0.9999,AUTHORITY["EPSG",
"8805"]],PARAMETER["False easting",0,AUTHORITY["EPSG","8806"]],PARAMETER["False northing",0,AUTHORITY["EPSG","880
7"]],UNIT["metre",1,AUTHORITY["EPSG","9001"]],AXIS["X",NORTH],AXIS["Y",EAST],AUTHORITY["EPSG","2448"]]
               DESCRIPTION: NULL
  ... 中略 ...
*************************** 2036. row ***************************
                SRS_NAME: WGS 84
                  SRS_ID: 4326
            ORGANIZATION: EPSG
ORGANIZATION_COORDSYS_ID: 4326
              DEFINITION: GEOGCS["WGS 84",DATUM["World Geodetic System 1984",SPHEROID["WGS 84",6378137,298.257223
563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","6326"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degr
ee",0.017453292519943278,AUTHORITY["EPSG","9122"]],AXIS["Lat",NORTH],AXIS["Lon",EAST],AUTHORITY["EPSG","4326"]]
               DESCRIPTION: NULL
  ... 中略 ...
5238 rows in set (0.0012 sec)
MySQL  localhost:33060+ ssl  SQL >
```

対応座標系の格納VIEW

座標系の名前

SRID

座標系定義、PROJCSで始まるのは投影座標系

デフォルトの座標の並び順 (X,Y)の順番

座標系の名前

SRID

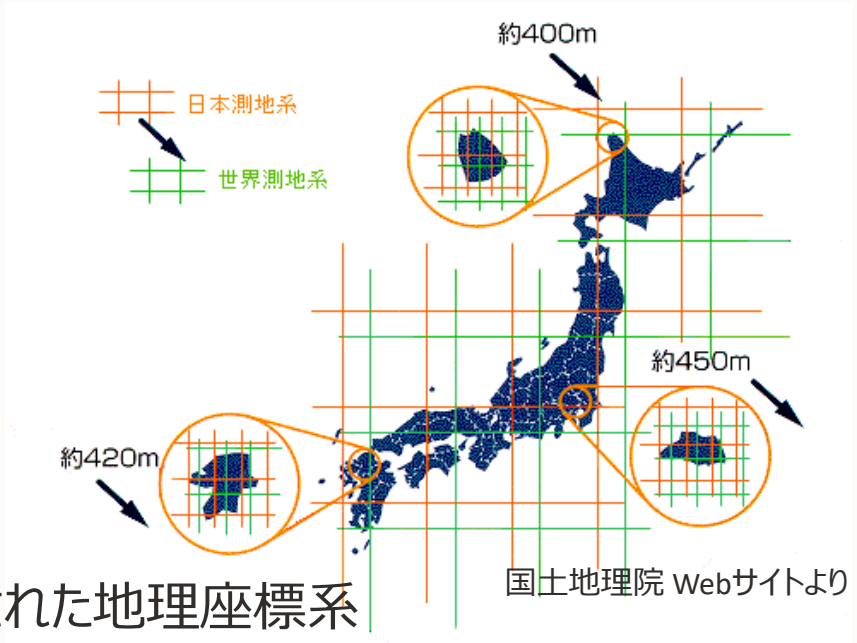
座標系定義、GEOGCSで始まるのは地理座標系

デフォルトの座標の並び順 (緯度,経度)の順番

5238座標系が登録されている

（内訳：地理座標系545種+投影座標系4692種+SRID=0）

覚えるべき座標系：地理座標系

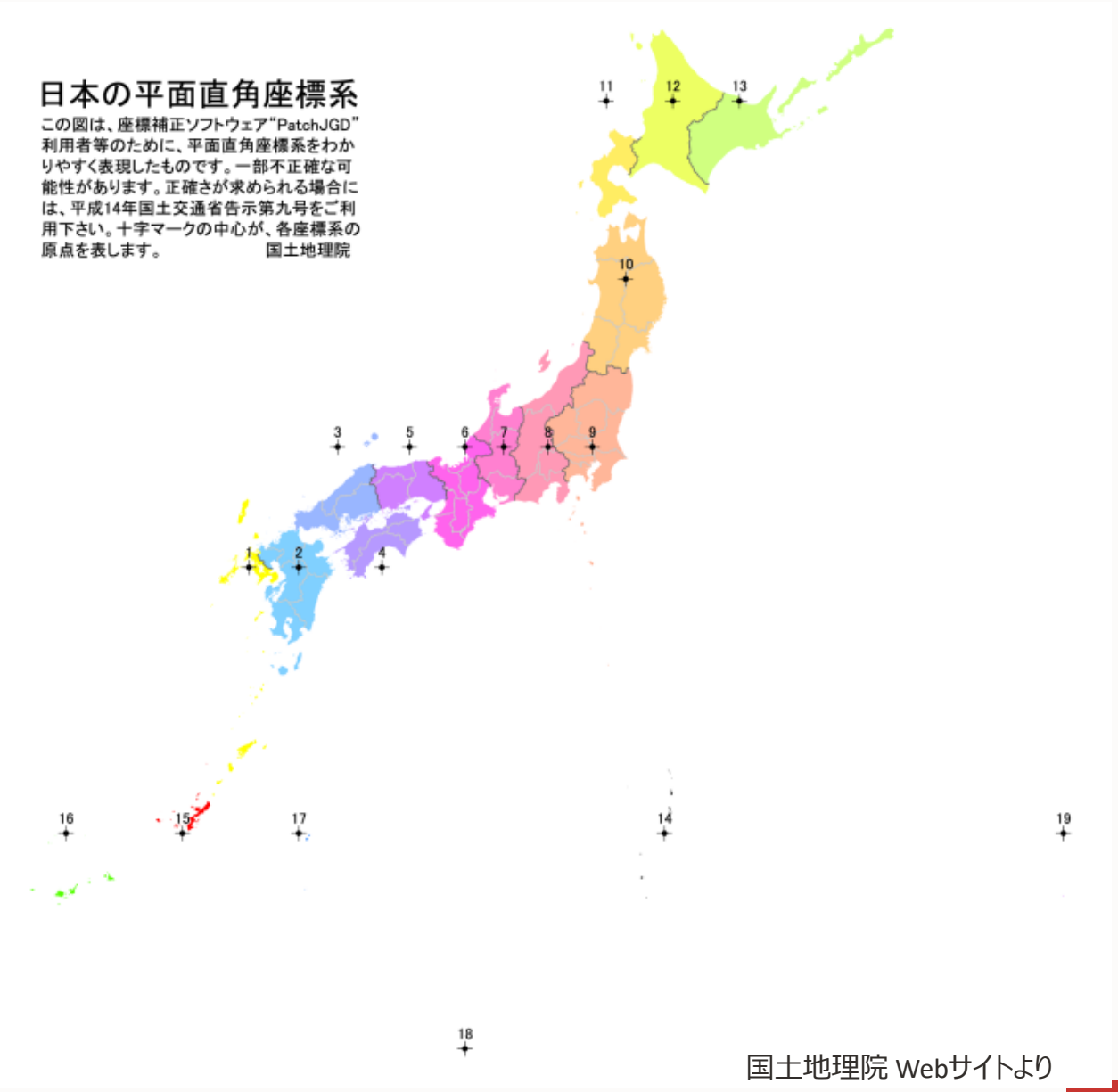


国土地理院 Webサイトより

- WGS84（世界測地系）SRID 4326
  - GPSなどで用いられる地理座標系、事実上の世界標準
- JGD2011（日本測地系2011）SRID 6668
  - 東日本大震災以降の歪みに対応するため、2011年新たに定義された地理座標系
  - 基本的にはJGD2000と同様の定義だが、東日本で数cmの差
  - MySQLでは座標定義のパラメータが不足しているため他座標との変換ができない
- JGD2000（日本測地系2000）SRID 4612
  - 世界標準であるWGS84とほぼ等しくなるよう2001年に定義された
- Tokyo（旧日本測地系）SRID 4301
  - 明治以降2000年まで使われていた、日本の標準地理座標系
  - 地球の中心位置や、近似に使っている回転楕円体の形状などがWGS84と異なる
  - 場所によって異なるが、WGS84やJGD系からおおむね400m～450mほどの位置ずれがある

覚えるべき座標系：投影座標系 (1)

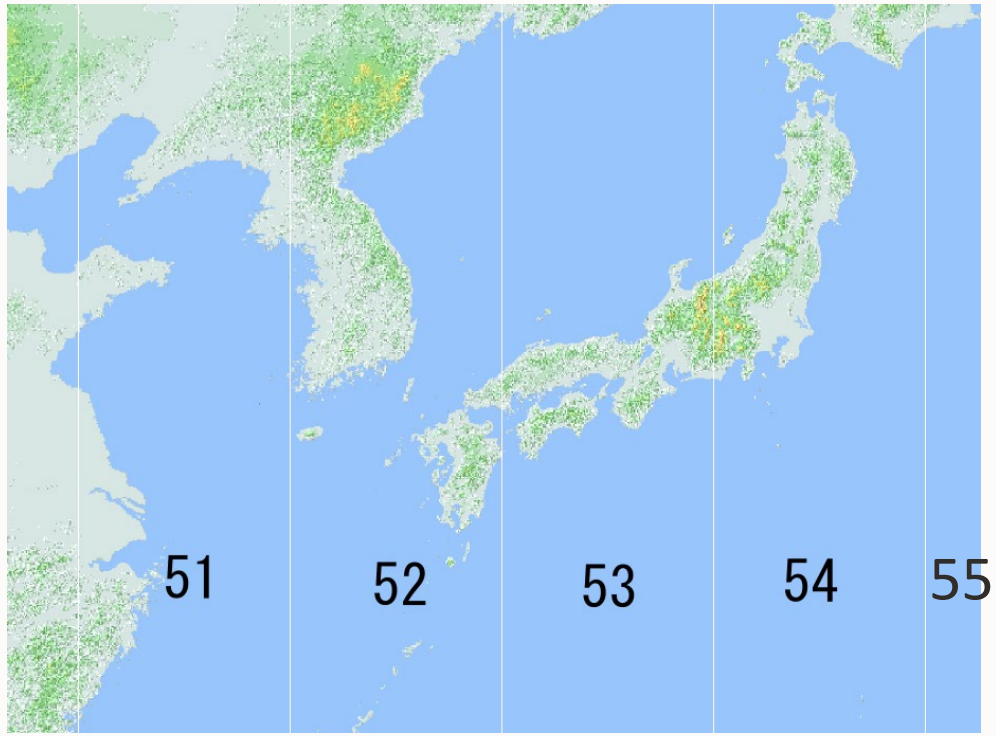
- 平面直角座標
  - Japan Plane Rectangular CS
  - 日本固有の座標系
  - 国土交通省告示第9号で定義
  - 全国を19のエリアに分割
  - 各エリアの原点からの北、東方向への距離で表す
  - 各エリアは自治体界を境界線とするよう定義
  - 測量図、都市計画図など大縮尺の地図で用いる



国土地理院 Webサイトより

覚えるべき座標系：投影座標系 (2)

- ユニバーサル横メルカトル（UTM）座標系
  - Universal Transverse Mercator
  - 経度6度ごとに区切った細長い帯に1～60の番号を付与
  - 西経180度から東回りに1から順に附番
  - 日本近郊はzone 51N～55Nでカバー
  - 地形図、地勢図などの中縮尺地図で用いる



Wikimedia Commons, CC BY-SA 4.0

覚えるべき座標系：投影座標系のSRID

	地理座標系	平面直角座標系					UTM座標系				
		I系	II系	…	XVIII	XIX	51N	52N	53N	54N	55N
世界測地系（WGS84）	4326	-	-	-	-	-	32651	32652	32653	32654	32655
日本測地系2011（JGD2011）	6668	6669	6670	…	6686	6687	6688	6689	6690	6691	6692
日本測地系2000（JGD2000）	4612	2443	2444	…	2460	2461	3097	3098	3099	3100	3101
旧日本測地系（Tokyo）	4301	30161	30162	…	30178	30179	3092	3093	3094	3095	3096

覚えるべき座標系：投影座標系 (3)

- Webメルカトル SRID 3857
  - 世界全体を正方形で表現するため、緯度約85度以上の北極や南極周辺の表現するのをあきらめた投影法
  - 地球を回転楕円体でなく球として近似した投影法、そのため球面メルカトルなどともいう
  - Google MapsのためにGoogleにより定義されたため、Googleメルカトルなどともいう
  - Webのスクロール地図（タイル地図）を実現する際の事実上の標準

OpenStreetMap and Contributors

SRIDを指定したテーブル作成とデータの挿入

```
MySQL  localhost:33060+ ssl  geo_test  SQL > CREATE TABLE test2(
                                        ->   id INT AUTO_INCREMENT PRIMARY KEY,
                                        ->   geom GEOMETRY SRID 4326
                                        -> );
Query OK, 0 rows affected (0.0272 sec)
MySQL  localhost:33060+ ssl  geo_test  SQL > INSERT INTO test2(geom) VALUES(ST_GeomFromText('POINT(35 135)', 432
6));
Query OK, 1 row affected (0.0059 sec)
MySQL  localhost:33060+ ssl  geo_test  SQL >
MySQL  localhost:33060+ ssl  geo_test  SQL > INSERT INTO test2(geom) VALUES(ST_GeomFromText('POINT(35 135)', 430
1));
ERROR: 3643: The SRID of the geometry does not match the SRID of the column 'geom'. The SRID of the geometry is 43
01, but the SRID of the column is 4326. Consider changing the SRID of the geometry or the SRID property of the col
umn.
MySQL  localhost:33060+ ssl  geo_test  SQL >
MySQL  localhost:33060+ ssl  geo_test  SQL > INSERT INTO test2(geom) VALUES(ST_GeomFromText('POINT(140 40)', 432
6));
ERROR: 3617: Latitude 140.000000 is out of range in function st_geomfromtext. It must be within [-90.000000, 90.00
0000].
MySQL  localhost:33060+ ssl  geo_test  SQL >
MySQL  localhost:33060+ ssl  geo_test  SQL > INSERT INTO test2(geom) VALUES(ST_GeomFromText('POINT(140 40)', 4326,
'axis-order=long-lat' );
Query OK, 1 row affected (0.0059 sec)
MySQL  localhost:33060+ ssl  geo_test  SQL >
```

世界測地系を指定した空間データ型カラム

同じSRIDを指定すると正常に挿入

違うSRIDは受け付けない

地理座標系のデフォルトの座標並びは緯度、経度

座標並び順をオプション指定できる

座標系間の座標変換

```
MySQL localhost:33060+ ssl SQL > SET @wgs84 = ST_GeomFromText('POINT(35 135)', 4326);
Query OK, 0 rows affected (0.0008 sec)
MySQL localhost:33060+ ssl SQL > SELECT ST_AsText(ST_Transform(@wgs84, 4301) ;
+---------------------------------------------+
| ST_AsText(ST_Transform(@wgs84, 4301))       |
+---------------------------------------------+
| POINT(34.996751669656696 135.00278101474856) |
+---------------------------------------------+
1 row in set (0.0041 sec)
MySQL localhost:33060+ ssl SQL > SELECT ST_AsText(ST_Transform(@wgs84, 4612) ;
+--------------------------------------+
| ST_AsText(ST_Transform(@wgs84, 4612)) |
+--------------------------------------+
| POINT(35 135)                        |
+--------------------------------------+
1 row in set (0.0008 sec)
MySQL localhost:33060+ ssl SQL > SELECT ST_AsText(ST_Transform(@wgs84, 6668) ;
ERROR: 3744: Transformation to SRID 6668 is not supported. The spatial reference system has no TOWGS84 clause.
MySQL localhost:33060+ ssl SQL > SELECT ST_AsText(ST_Transform(@wgs84, 2448) ;
+---------------------------------------------+
| ST_AsText(ST_Transform(@wgs84, 2448))       |
+---------------------------------------------+
| POINT(-91280.63976505134 -110481.75032585992) |
+---------------------------------------------+
1 row in set (0.0014 sec)
MySQL localhost:33060+ ssl SQL >
```

WGS84で空間データを定義

旧日本測地系に変換（地理座標）

日本測地系2000に変換
=> ほぼWGS84と同定義のため値は変わらない

日本測地系2011に変換
=> 必要なパラメータがない
ため変換できない

日本測地系2000
平面直角座標VI系に変換
（投影座標）

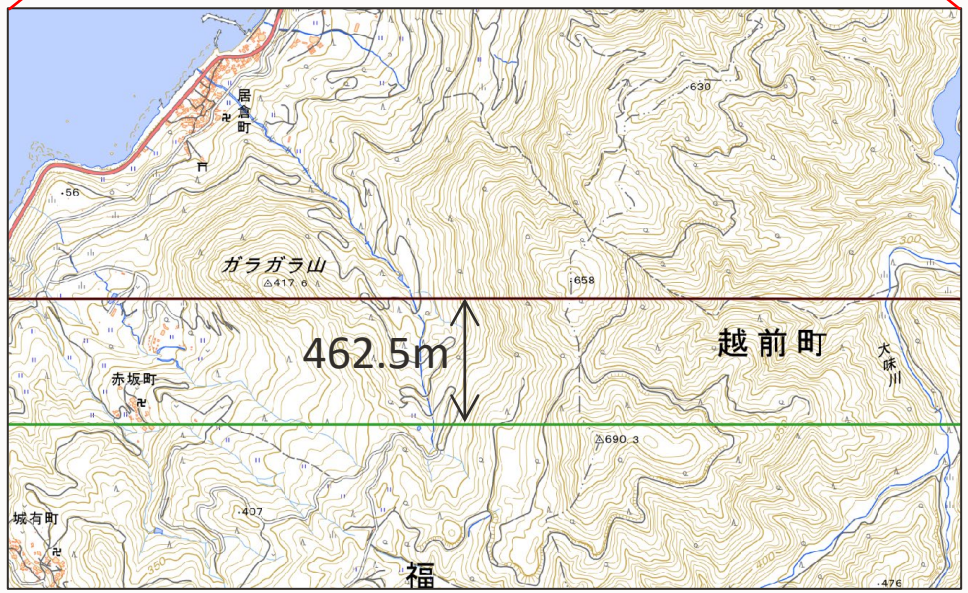
座標系の使い分け (1)

- データを取り込む場合：元データの用いる座標系を使う
  - GPSなどから得られた計測データ: WGS84
  - 測量成果のデータ（経緯度）: 時代によりTokyo、JGD2000、JGD2011を使い分け
  - 自治体などから出てくる地域の測量成果は、平面直角座標系も多い
- データを利用する場合：用途やルールに応じて、座標変換して利用
  - 投影座標系は、その投影法の特性を理解して利用
    - 距離が大きく歪む投影座標系でST_Distanceなどは意味がない
  - 地図に重ね合わせる場合は、その地図の投影座標系に変換
    - 地図APIやGISツール側で座標変換に対応している場合も多い
  - 処理に用いる座標系がルールで決められている場合もある
    - 測量に関わる処理は平面直角座標系を使うことが法規で定められている

座標系の使い分け (2)

- 座標系が違うと変わること
  - 基本的に「点」は、連続変換による誤差の蓄積を除いては変化しないと考えてよい
  - 「点」同士の関係性が変わる
    - 他の点の方向、角度、距離、点間を結ぶ辺の通る軌跡、etc.
    - 関係性の結果として定義される点は変わる　例：辺と辺の交点の位置など
  - 例：WGS84、平面直角座標系、Webメルカトルの比較（右図）
    - WGS84とWebメルカトル間の中点での誤差462.5m
    - WGS84と平面直角VI系間の中点での誤差0m（計算限界以下）
  - ルールがない限りはWGS84で計算すればほぼ問題ない
    - 各座標に適した計算を自動で切り替えるMySQLの優位点

地理院地図、平面直角座標VI系



地理院地図、平面直角座標VI系

5. 空間演算関数

距離を求める関数例

- ST_Distance　2地点間の距離を計算
  - 構文　ST_Distance(g1, g2 [, unit])
  - 地理座標系では回転楕円体面上での距離、投影座標系では座標系上での距離を計算
  - unitに指定できる単位は、以下のSQLで確認可能

```
MySQL  localhost:33060+ ssl  SQL > SELECT * FROM INFORMATION_SCHEMA.ST_UNITS_OF_MEASURE;
+-------------------------------------+-----------+---------------------+-------------+
| UNIT_NAME                           | UNIT_TYPE | CONVERSION_FACTOR   | DESCRIPTION |
+-------------------------------------+-----------+---------------------+-------------+
| metre                               | LINEAR    |                   1 |             |
| millimetre                          | LINEAR    |               0.001 |             |
| centimetre                          | LINEAR    |                0.01 |             |
| German legal metre                  | LINEAR    |        1.0000135965 |             |
| foot                                | LINEAR    |              0.3048 |             |
| US survey foot                      | LINEAR    | 0.30480060960121924 |             |
  ... 中略 ...
| kilometre                           | LINEAR    |                1000 |             |
  ... 中略 ...
| British yard (Sears 1922 truncated) | LINEAR    |            0.914398 |             |
| British foot (Sears 1922 truncated) | LINEAR    | 0.30479933333333337 |             |
| British chain (Sears 1922 truncated)| LINEAR    |           20.116756 |             |
+-------------------------------------+-----------+---------------------+-------------+
47 rows in set (0.0189 sec)
```

ST_Distance の動作例

```
MySQL  localhost:33060+ ssl  SQL > SET @aoyama = ST_GeomFromText('POINT(35.67133 139.71857)', 4326);
Query OK, 0 rows affected (0.0009 sec)
MySQL  localhost:33060+ ssl  SQL > SET @daiba = ST_GeomFromText('POINT(35.62854 139.77783)', 4326);
Query OK, 0 rows affected (0.0006 sec)
MySQL  localhost:33060+ ssl  SQL > SELECT ST_Distance(@aoyama, @daiba, 'kilometre') As KM;
+------------------+
| KM               |
+------------------+
| 7.16525638596044 |
+------------------+
1 row in set (0.0007 sec)
MySQL  localhost:33060+ ssl  SQL > SELECT ST_Distance(ST_Transform(@aoyama, 3857), ST_Transform(@daiba, 3857), 'k
ilometre') As KM;
+--------------------+
| KM                 |
+--------------------+
| 8.824956786338664  |
+--------------------+
1 row in set (0.0013 sec)
MySQL  localhost:33060+ ssl  SQL > SELECT ST_Distance(ST_Transform(@aoyama, 2451), ST_Transform(@daiba, 2451), 'k
ilometre') As KM;
+--------------------+
| KM                 |
+--------------------+
| 7.164538605932254  |
+--------------------+
1 row in set (0.0017 sec)
MySQL  localhost:33060+ ssl  SQL >
```

青山の経緯度 WGS84

お台場の経緯度 WGS84

単位はkm

地理座標系上での距離計算
およそ7.165km

Webメルカトルへ変換
しての距離計算

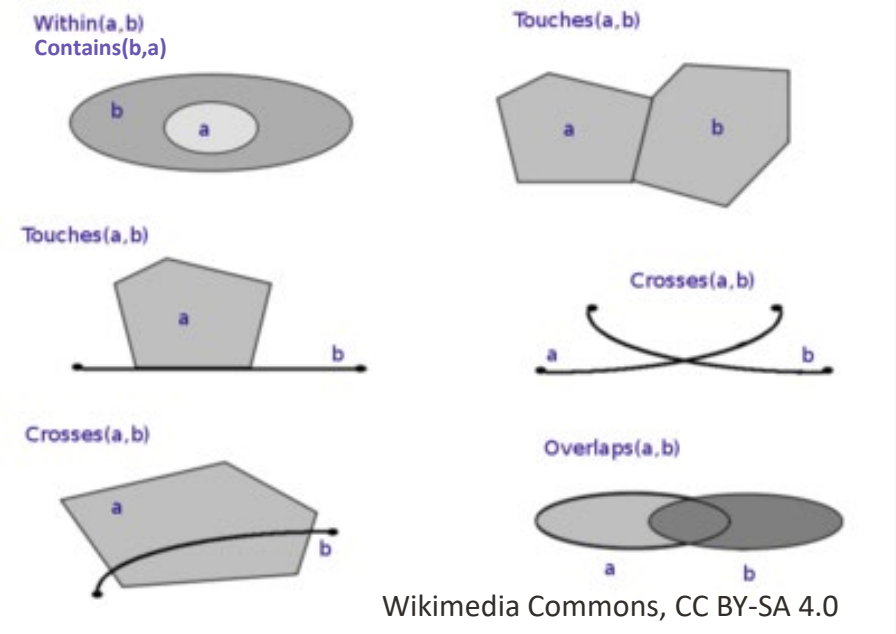
投影座標系上での距離計算
Webメルカトル地図は距離が不正確なため
WGS84上での計算結果と大きく異なる

JGD2000 平面直角IX系へ
変換しての距離計算

投影座標系上での距離計算
平面直角座標系は比較的距離が正確なため
WGS84上での計算結果とかなり近い

幾何的関係を求める関数例

- ST_Contains　空間オブジェクトが他の空間オブジェクトの中に含まれるかの判定
  - 構文　ST_Contains(g1, g2)
  - g1 が g2 を完全に含んでいるかどうかを判定、含む場合は1、含まなければ0を返す
- その他の関数
  - ST_Within、ST_Touches、ST_Crosses、ST_Overlaps等



Wikimedia Commons, CC BY-SA 4.0

ST_Contains の動作例

```
MySQL localhost:33060+ ssl SQL > SET @point = ST_GeomFromText('POINT(40.00001 135)', 4326);
Query OK, 0 rows affected (0.0009 sec)
MySQL localhost:33060+ ssl SQL > SET @polygon = ST_GeomFromText('POLYGON((40 130, 40 140, 30 140, 30 130, 40 13
0))', 4326);
Query OK, 0 rows affected (0.0006 sec)
MySQL localhost:33060+ ssl SQL > SELECT ST_Contains(@polygon, @point) As Ellipsoid_Contains;
+--------------------+
| Ellipsoid_Contains |
+--------------------+
|                  1 |
+--------------------+
1 row in set (0.0009 sec)
MySQL localhost:33060+ ssl SQL > SELECT ST_Contains(ST_Transform(@polygon, 3857), ST_Transform(@point, 3857)) A
s Cartesian_Contains;
+--------------------+
| Cartesian_Contains |
+--------------------+
|                  0 |
+--------------------+
1 row in set (0.0077 sec)
MySQL localhost:33060+ ssl SQL >
```

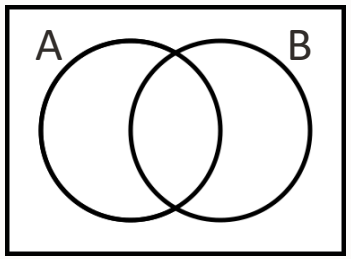
点とポリゴンをWGS84で定義

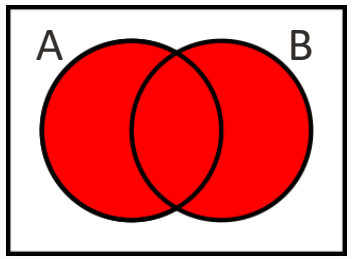
点がポリゴンの中に存在するかを確認
WGS84で判定すると内部と判定される

同じ点とポリゴンをWebメルカトル座標系に
変換した上で存在するかを確認
Webメルカトルで判定すると外部と判定される

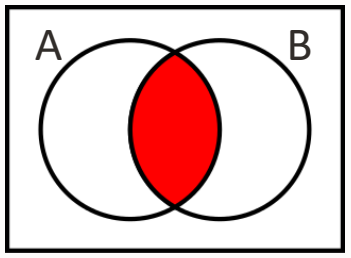
幾何演算をおこなう関数例

- ST_Union　2つの空間オブジェクトを合成した新しい空間オブジェクトを生成
  - 構文　ST_Union(g1, g2)
  - 集約関数的な動作はできない
- その他の関数
  - ST_Intersection、ST_Difference、ST_SymDifference等

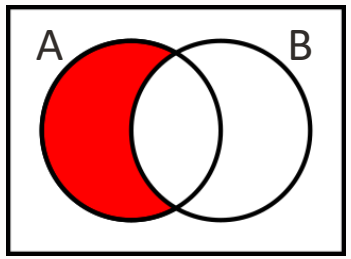




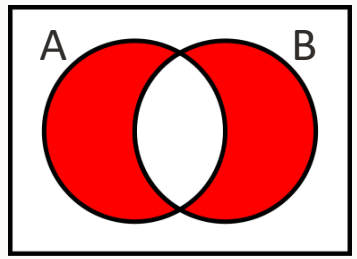
Union



Intersection



Difference
(From B)



SymDifference

- 他にも様々な関数が存在
  - 空間関数のリファレンスをみてください

6. 空間検索と空間インデックス

空間検索と空間インデックス

- SELECT文のWHERE条件に対し、幾何的関係を求める関数を指定し検索
  - 例: ポリゴン@polygonの範囲に含まれる空間カラム要素を検索
    SELECT AsText(geom) FROM geo_table WHERE ST_Contains(@polygon, geom);

- 検索を高速化するためのインデックス構築
  - SPATIALキーワードを付与してインデックス構築
  - CREATE TABLEの例:
    CREATE TABLE geo_table (geom GEOMETRY NOT NULL SRID 4326, SPATIAL INDEX(geom));
  - ALTER TABLEの例:
    ALTER TABLE geo_table ADD SPATIAL INDEX(geom);
  - CREATE INDEXの例:
    CREATE SPATIAL INDEX g_idx ON geo_table (geom);

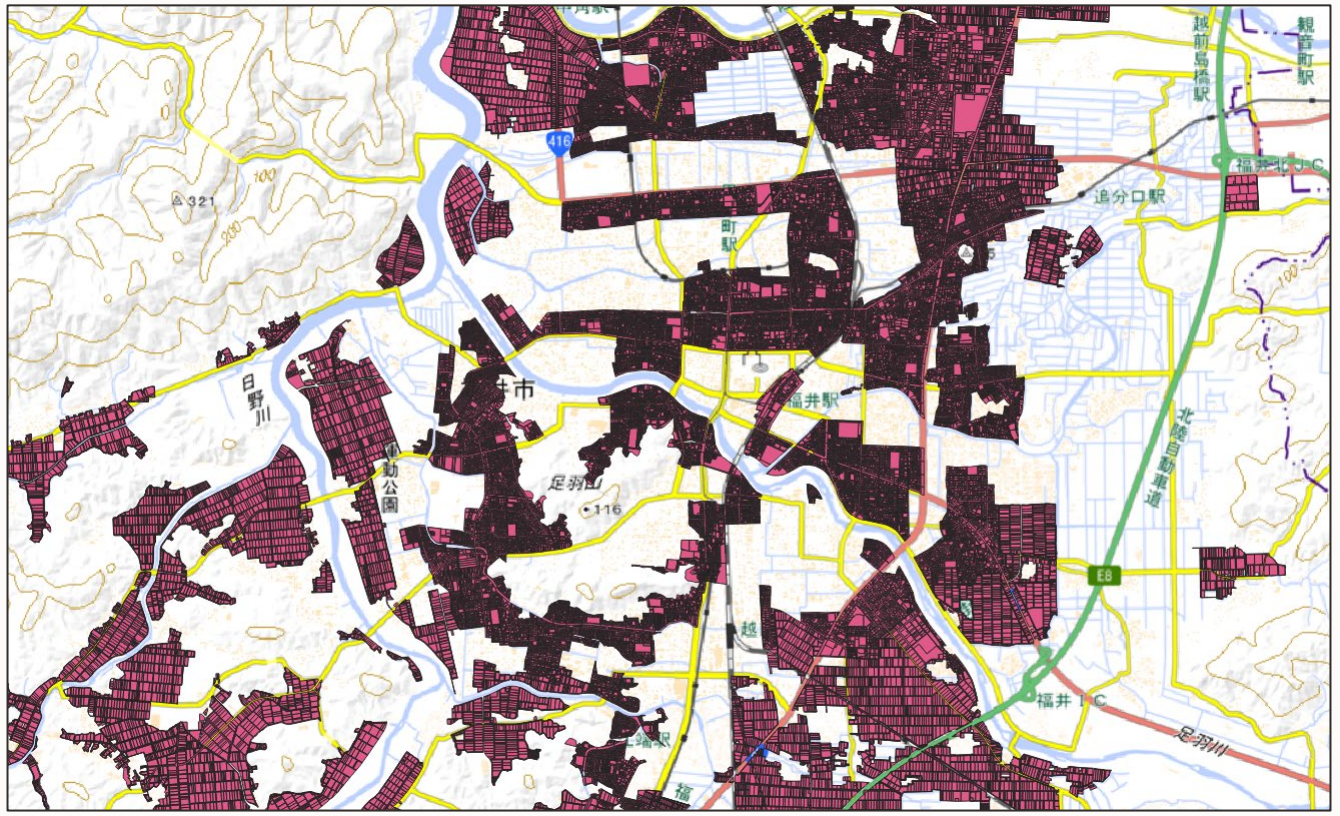
空間検索環境の準備 (1) – 法務省登記所備付地図データ

GeoJSONダウンロードには
利用者登録とログインが必要

- 法務省登記所備付地図データ
  - 法務局の登記所で用いられる、不動産登記で付与される“地番”の情報を持つ地図データ
  - 2023年1月23日、突然無償公開データとなって話題となった
  - 正確な地図に重ね合わせ出来ない「任意座標系」と、重ね合わせられる「公共座標系」がある
  - 扱いにくいデータ形式 => G空間情報センターで、扱いやすいGeoJSON形式に

空間検索環境の準備 (2) – 法務省登記所備付地図データ（福井市）

- 福井市のデータ例　（18201__6_r.geojson）
  - 地番を持つ個々の領域（筆と呼ばれる）がデータ化されている
  - 隙間があるのは任意座標系地域



地理院地図 平面直角座標VI系 &
法務省登記所備付地図データ

空間検索環境の準備 (3) – GeoJSONとデータベースのスキーマ

GeoJSON

```
{
  "type": "FeatureCollection",
  "name": "18201_福井市_公共座標6系_筆R",
  "crs": { ... },
  "features": [
    { "type": "Feature",
      "properties": {
        "ID": "H000000001",
         ...
      },
      "geometry": {
        "type": "Polygon",
        "coordinates": [ [
          [ 136.231643178405932, 36.05641148163491 ],
           ...
        ] ]
      }
    },
     ...
  ]
}
```

筆の属性

筆のポリゴン

MySQLの空間テーブル

```
CREATE TABLE geo_table (
  id int unsigned NOT NULL AUTO_INCREMENT,
  geometry geometry NOT NULL SRID 4326,
  property json NOT NULL,
  PRIMARY KEY (id),
  SPATIAL KEY idx_geometry (geometry)
) ENGINE=InnoDB
```

空間インデックス

空間検索環境の準備 (4) – データを読み込むためのNode.jsコード

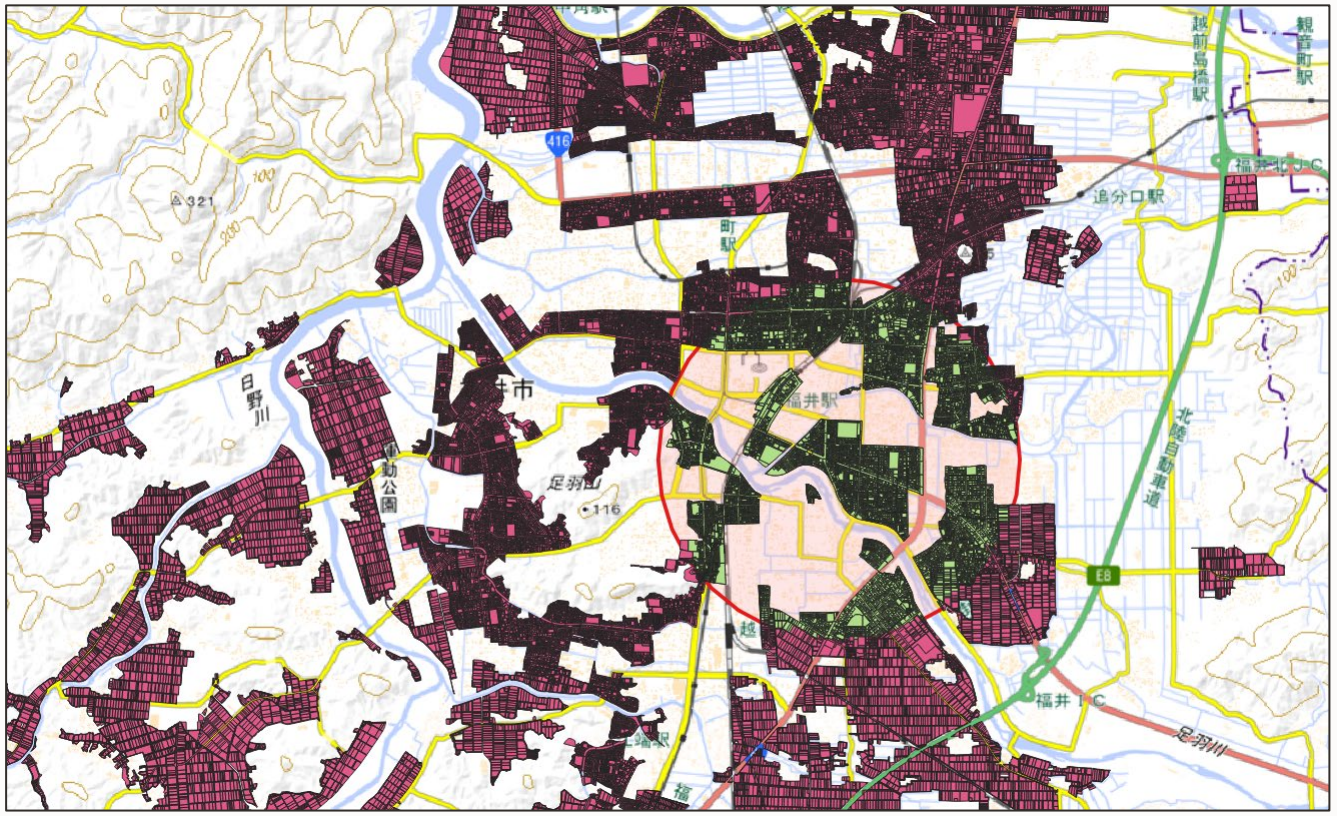
```
import mysqlx from "@mysql/xdevapi";
import fs from "fs-extra";
const json = fs.readJSONSync("./18201__6_r.geojson");

const session = await mysqlx.getSession({
  host: 'localhost', port: 33060, user: 'root', password: 'PASSWORD'
});
await session.sql('use geo_test').execute();
const awaiter = json.features.reduce((prev, feature) => {
  return prev.then(() => {
    const sql = `INSERT INTO geo_table (geometry, property) VALUES (
      ST_GeomFromGeoJSON('${JSON.stringify(feature.geometry)}', 1, 4326),
      '${JSON.stringify(feature.properties)}'
    );`;
    return session.sql(sql).execute();
  });
}, Promise.resolve());

await awaiter;
console.log("Finished");
```

各筆のgeometry属性を
ST_GeomFromGeoJSON関数で
空間オブジェクト化しgeometry
カラムに挿入

各筆のproperties属性を
JSON型であるproperty
カラムに挿入

空間検索の実施

- ST_Contains関数で図形の内部に含まれる空間オブジェクトを検索
  - (136.23175, 36.05601)を中心とした2kmの円を生成（ST_Buffer関数利用）し検索
  - 空間インデックスをオン/オフして効率を比較



地理院地図 平面直角座標VI系 &
法務省登記所備付地図データ

空間検索の動作結果

```
MySQL  localhost:33060+ ssl  geo_test  SQL > SELECT COUNT(*) FROM geo_table;
+-----------+
| COUNT(*)  |
+-----------+
|    118158 |
+-----------+
1 row in set (0.4059 sec)
MySQL  localhost:33060+ ssl  geo_test  SQL > ALTER TABLE geo_table ALTER INDEX idx_geometry invisible;
Query OK, 0 rows affected (0.0098 sec)

Records: 0  Duplicates: 0  Warnings: 0
MySQL  localhost:33060+ ssl  geo_test  SQL > SELECT ST_AsWKT(geometry), property FROM geo_table WHERE ST_Contains
(ST_Buffer(ST_GeomFromText('POINT(36.05601 136.23175)', 4326), 2000), geometry);
 ... 中略 ...
21522 rows in set (0.0042 sec)
MySQL  localhost:33060+ ssl  geo_test  SQL > EXPLAIN ANALYZE SELECT ST_AsWKT(geometry), property FROM geo_table W
HERE ST_Contains(ST_Buffer(ST_GeomFromText('POINT(36.05601 136.23175)', 4326), 2000), geometry);
 ... 中略 ...
| -> Filter: st_contains(<cache>(st_buffer(st_geomfromtext('POINT(36.05601 136.23175)',4326),2000)),geo_table.`geo
metry`)  (cost=14501 rows=106739) (actual time=0.276..21447 rows=21522 loops=1)
    -> Table scan on geo_table  (cost=14501 rows=106739) (actual time=0.0307..2394 rows=118158 loops=1)
 |
 ... 中略 ...
1 row in set (22.8109 sec)
MySQL  localhost:33060+ ssl  geo_test  SQL >
```

空間オブジェクト数は118,158件

空間インデックスを無効化

ST_Buffer関数で、中心点から2km半径の円を生成し、それと重なる筆を検索

インデックスが使われない全件テーブルスキャン、かかった時間は23秒弱

空間検索の動作結果（地理座標系空間インデックス利用）

```
MySQL  localhost:33060+ ssl  geo_test  SQL > ALTER TABLE geo_table ALTER INDEX idx_geometry visible;
Query OK, 0 rows affected (0.0098 sec)

Records: 0  Duplicates: 0  Warnings: 0
MySQL  localhost:33060+ ssl  geo_test  SQL > SELECT ST_AsWKT(geometry), property FROM geo_table WHERE ST_Contains
(ST_Buffer(ST_GeomFromText('POINT(36.05601 136.23175)', 4326), 2000), geometry);
 ... 中略 ...
21522 rows in set (0.0182 sec)
MySQL  localhost:33060+ ssl  geo_test  SQL > EXPLAIN ANALYZE SELECT ST_AsWKT(geometry), property FROM geo_table W
HERE ST_Contains(ST_Buffer(ST_GeomFromText('POINT(36.05601 136.23175)', 4326), 2000), geometry);
 ... 中略 ...
| -> Filter: st_contains(<cache>(st_buffer(st_geomfromtext('POINT(36.05601 136.23175)',4326),2000)),geo_table.`geo
metry`)  (cost=3457 rows=5325) (actual time=9.54..5089 rows=21522 loops=1)
    -> Index range scan on geo_table using idx_geometry over (geometry unprintable_geometry_value)  (cost=3457 row
s=5325) (actual time=0.0941..600 rows=27131 loops=1)
 |
 ... 中略 ...
1 row in set (6.4161 sec)
MySQL  localhost:33060+ ssl  geo_test  SQL >
```

空間インデックスを有効化

前ページと同じ条件の検索

インデックスが使われ、全件118,158件のスキャンから27,131件のスキャンに縮小、かかった時間は6秒強

⇒ 地理座標系での空間インデックスにより4倍程度の高速化

空間検索の動作結果（投影座標系空間インデックス利用）

```
MySQL  localhost:33060+ ssl  geo_test  SQL > ALTER TABLE geo_table ADD COLUMN geometry2448 GEOMETRY GENERATED ALW
AYS AS (ST_Transform(geometry, 2448)) STORED SRID 2448 NOT NULL;
Query OK, 118158 rows affected (4 min 25.5102 sec)

Records: 118158  Duplicates: 0  Warnings: 0
MySQL  localhost:33060+ ssl  geo_test  SQL > CREATE SPATIAL INDEX idx_geometry2448 ON geo_table (geometry2448);
Query OK, 0 rows affected (29.5763 sec)

Records: 0  Duplicates: 0  Warnings: 0
MySQL  localhost:33060+ ssl  geo_test  SQL > SELECT ST_AsWKT(geometry), property FROM geo_table WHERE ST_Contains
(ST_Transform(ST_Buffer(ST_GeomFromText('POINT(36.05601 136.23175)', 4326), 2000), 2448), geometry2448);
 ... 中略 ...
21522 rows in set (0.0117 sec)
MySQL  localhost:33060+ ssl  geo_test  SQL > EXPLAIN ANALYZE SELECT ST_AsWKT(geometry), property FROM geo_table W
HERE ST_Contains(ST_Transform(ST_Buffer(ST_GeomFromText('POINT(36.05601 136.23175)', 4326), 2000), 2448), geometry
2448);
 ... 中略 ...
| -> Filter: st_contains(<cache>(st_transform(st_buffer(st_geomfromtext('POINT(36.05601 136.23175)',4326),2000),24
48)),geo_table.geometry2448)  (cost=5971 rows=4975) (actual time=4.59..2588 rows=21522 loops=1)
    -> Index range scan on geo_table using idx_geometry2448 over (geometry2448 unprintable_geometry_value)  (cost=
5971 rows=4975) (actual time=0.168..340 rows=27111 loops=1)
 |
 ... 中略 ...
1 row in set (3.9910 sec)
MySQL  localhost:33060+ ssl  geo_test  SQL >
```

WGS84座標から、平面直角VI系座標をGENERATEDカラムで作成
（空間インデックスには STORED NOT NULLが必要）

平面直角VI系座標に空間インデックス付与

同じ条件の検索を平面直角VI系座標上で行う

インデックスが使われ、27,111件のスキャン、投影座標系検索でかかった時間は4秒弱

⇒ 投影座標系での空間インデックスにより6倍程度の高速化
GENERATEDカラムの利用により、列挿入は負荷増

7. MySQLの利用方法

柔軟なMySQLの利用方法

MySQLサーバーは全て共通のソースコードのためハイブリッド構成も可能

MySQLを自社で運用管理

オンプレミスでのMySQL

- バージョン選択や構成を最も柔軟に選択可能

IaaS上でのMySQL

- OCIのマーケットプレイスのイメージから簡単に環境構築

商用版MySQL

- コミュニティ版に加え、サポートやセキュリティに優れた商用版も

MySQLのマネージドサービス

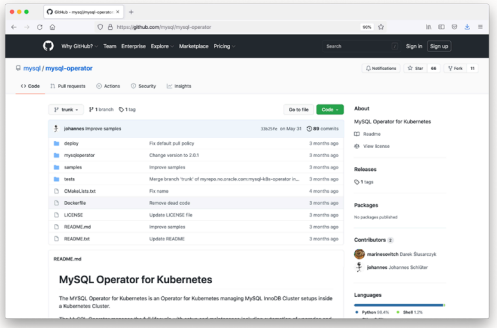
MySQL HeatWave Database Service

- MySQLチームが100%開発・提供するクラウド・サービス
- Amazon RDS (MySQL)の1/3以下のコスト
- データ損失なし、自動フェイルオーバーの高可用性機能をマネージドサービスで提供

クラウドネイティブなMySQL

MySQL Operator for k8s

- MySQL サーバーをKubernetes上に構築し運用管理



MySQL :: MySQL Operator for Kubernetes テクニカルアップデート

いずれの利用方法でもMySQL開発チームと連携した専門部隊によるサポートサービスをご利用いただけます※

※コミュニティ版を除く

Oracle Premier Support for MySQL

- 最大のMySQLのエンジニアリングおよびサポート組織
- MySQL開発チームによるサポート
- 29言語で世界クラスのサポートを提供
- メンテナンス・リリース、バグ修正、パッチ、アップデートの提供
- 24時間x365日サポート
- MySQL コンサルティング・サポート

Get immediate help for any MySQL issue, plus expert advice

開発チームと一体となったサポートサービス

⇒ 商用版MySQL サーバー及びMySQL HeatWave Database Serviceにより提供

年間サブスクリプション 74.9万円 (1サーバーあたり)

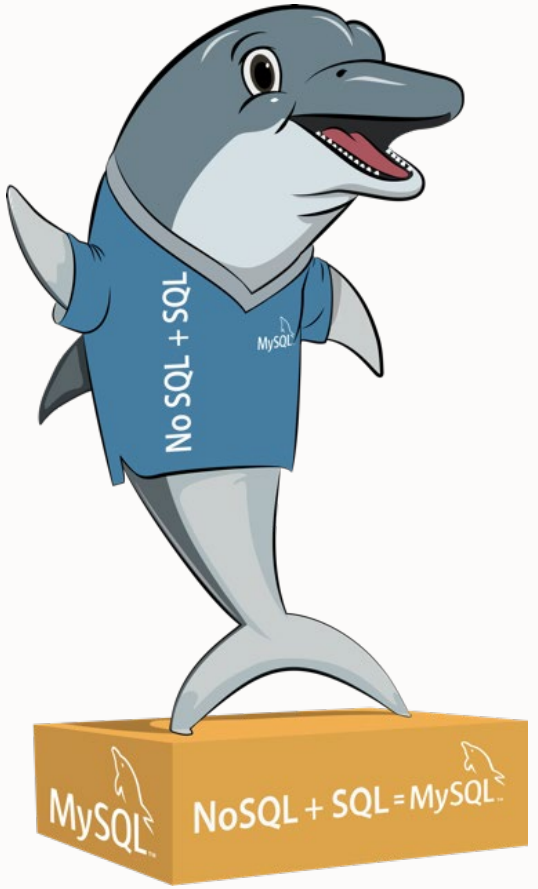
MySQLの最新情報配信

MySQLホームページ
http://www.mysql.com/jp

MySQL イベント
http://www.mysql.com/jp/news-and-events/

MySQLニュースレター 英語版&日本語版（月刊）
https://www.mysql.com/jp/news-and-events/newsletter/

MySQL Twitter 日本語公式アカウント
@mysql_jp



本セッションのまとめ

MySQL の GIS 機能

- 空間データを検索するには、GIS機能が必要
- MySQLのGIS機能の特徴
  - 8.0以降、地理座標系に対応 => 地球が丸いことを覚えた！
  - GEOMETRY型のみで、地理座標系と投影座標系の処理を切り替えてくれる
  - 多くの空間関数が、両方の座標系に対応
  - 用途による座標系の使い分け
- 空間検索
  - 空間演算関数を組み合わせて検索条件とする
  - 空間インデックスの利用による検索高速化
- MySQLの利用法
  - オンプレミス、マネージドサービス、k8sなど幅広い選択肢
  - 開発ベンダーによるサポートサービスが存在

Appendix

MySQL コミュニティ

- MySQL コミュニティの紹介
- MySQL への貢献
  - Oracle Contribution Agreement (OCA)
- MySQL無償認証制度

MySQL コミュニティの紹介

- MySQL コミュニティへの貢献プロセスの運営
- MySQL ユーザーグループへの支援 https://dev.mysql.com/community/mug/
- 全世界でのサードパーティによるカンファレンスやイベントへの支援や参加 https://dev.mysql.com/community/
- 教育ビデオの作成
  - MySQL 短編動画 (MySQL Shorts)
  - MySQL 入門編シリーズ (MySQL 101 for Biginners)
    - https://www.youtube.com/@mysql

- MySQL RockStar プログラム
  - MySQLの利用促進に最も精力的に取り組んだ MySQL コミュニティ・メンバーへの表彰
  - 第1回: https://blogs.oracle.com/mysql/post/mysql-rockstars-2022
- MySQL ACE プログラム
  - MySQL プロジェクトでの ACE プログラムの運営
  - https://ace.oracle.com/pls/apex/ace_program/r/oracle-aces/home

MySQL への貢献

- MySQL オープンソースプロジェクトのコントリビューターコミュニティへの参加: https://forums.oracle.com/ords/apexds/post/contributing-code-to-mysql-8037
- コントリビュータになるために持つべきこと
  - MySQL の機能を変更/修正したい、あるいは新しい機能を追加したいといった要望
  - MySQL ソースコードのダウンロード http://dev.mysql.com/downloads/
  - bugs.mysql.com のアカウント http://bugs.mysql.com or
  - 有効な GitHub アカウント https://github.com
- Oracle Contribution Agreement (OCA) への署名 https://oca.opensource.oracle.com/
  - OCAは、コントリビューターとオラクルの両方を法的攻撃から保護する短い法的契約です。OCAに署名することにより、コントリビューターはオラクルがコントリビューターのコードをオラクル・ソフトウェアで使用することが法的に許可されていること、およびコントリビューターの知る限りにおいて、そのコードに特許的な問題がないことに同意することになります。

MySQL 無償認証制度

- MySQL コミュニティチームは、Oracle University および Oracle Academy と協力し、mylearn.oracle.com を介して、2ヶ月間の指定期間内に使用できる無料のトレーニングバウチャー/クレジットを受講者に提供します。
- ご興味のある方は、以下についての詳細をお知らせくだされば、MySQL コミュニティから連絡いたします。
  - 名前
  - 姓
  - Email アドレス
  - 居住国
- **https://education.oracle.com/**

連絡先

- **MySQL コミュニティとのコンタクト先一覧:**
- MySQL コミュニティページ, https://dev.mysql.com/community/
- MySQL Slack, https://mysqlcommunity.slack.com
- The Oracle MySQL ブログ, https://blogs.oracle.com/mysql/
- The Oracle MySQL Japan ブログ, https://blogs.oracle.com/mysql-jp/
- Planet MySQL, https://planet.mysql.com/
- LinkedIn, https://www.linkedin.com/groups/60715/
- ブログ, https://lefred.be/
- MySQL フォーラム, http://lists.mysql.com/
- ディスカッションフォーラム, http://forums.mysql.com


ORACLE