

ADAMS

Advanced Data mining And Machine learning System

Module: adams-maps



Peter Reutemann
Dale Fletcher

December 20, 2017

©2012-2013



THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato



Except where otherwise noted, this work is licensed under
<http://creativecommons.org/licenses/by-sa/4.0/>

Contents

1	Flow	5
2	Data	7
3	Databases	9
3.1	Postgresql	9
3.1.1	Installation	9
3.1.2	Data types	9
3.2	MySQL	10
	Bibliography	11

Chapter 1

Flow

The *adams-maps* module adds basic GIS-capability (geographic information system) to the ADAMS framework. This is done mainly in the form of GPS support: data types and spreadsheet support.

Chapter 2

Data

The following spreadsheet readers are available:

- *ArcInfoASCIIGridReader* – reads files in ARC/INFO ASCII GRID format[2].

The following spreadsheet object handlers are available:

- *GPSDecimalDegrees*
- *GPSDecimalMinutes*
- *GPSDecimalSeconds*

The following conversions are available:

- *SpreadSheetToKML* – turns a spreadsheet into KML[3].

Chapter 3

Databases

The following sections describe how you can utilize the GIS capabilities of various database systems.

3.1 Postgresql

In order to make use of GIS functionality, you need to install the *postgis* extension[5] for PostgreSQL[4].

3.1.1 Installation

For any new database, you need to install the extensions first:

```
CREATE EXTENSION postgis;
```

3.1.2 Data types

In order to make use of GPS coordinates in a table, e.g., for calculating distances between them, you need to create a combined column. The following example queries create a combined column *lon_lat* from the columns *lon* and *lat* of the *some_table* table. As data type, SRID 4269[6] (also called NAD 83) is used.

First, you need to create the column *lon_lat* and define it as data type *POINT*:

```
SELECT AddGeometryColumn(  
    'public', 'some_table', 'lon_lat', 4269, 'POINT', 2);
```

Second, you need to fill it with data, using the *ST_SetSRID* function, taking points generated from the *lon* and *lat* columns:

```
UPDATE some_table  
SET lon_lat = ST_SetSRID(ST_Point(lon, lat), 4269);
```

Having done this, you can execute queries, e.g., retrieving records that have a distance of less than 50km to the GPS coordinates of longitude of 28.136015 and latitude of -14.613297, ordered by increasing distance. For distance calculation the *ST_distance_sphere* method is used:

```
SELECT *, ST_distance_sphere(  
    ST_GeomFromText('POINT(28.136015 -14.613297)', 4269),  
    ST_GeomFromText(ST_asText(lon_lat), 4269)) AS dist  
FROM  
    some_table  
WHERE ST_distance_sphere(  
    ST_GeomFromText('POINT(28.136015 -14.613297)', 4269),  
    ST_GeomFromText(ST_asText(lon_lat), 4269)) < 50 * 1000.0  
ORDER BY  
    dist ASC
```

NB: These distance calculations are computationally quite expensive and, if possible, you should have indices on the *lat*, *long* and *lon_lat* columns, as well as restricting the window for the longitude and latitude that you are performing this query on.

3.2 MySQL

Spatial extensions in MYSQL¹

¹<https://dev.mysql.com/doc/refman/5.6/en/spatial-extensions.html>

Bibliography

- [1] *ADAMS* – Advanced Data mining and Machine learning System
<https://adams.cms.waikato.ac.nz/>
- [2] *Esri Grid* – a raster GIS file format deveoped by Esri.
https://en.wikipedia.org/wiki/Esri_grid
- [3] *Keyhole Markup Language* – an XML notation for expressing geographic annotation and visualization within Internet-based, two-dimensional maps and three-dimensional Earth browsers.
http://en.wikipedia.org/wiki/Keyhole_Markup_Language
- [4] *PostgreSQL* – a powerful, open source object-relational database system.
<http://www.postgresql.org/>
- [5] *PostGIS* – a spatial database extender for PostgreSQL object-relational database. It adds support for geographic objects allowing location queries to be run in SQL.
<http://postgis.net/>
- [6] *SRID 4269* – or NAD 83 (North American Datum).
<http://spatialreference.org/ref/epsg/4269/>
- [7] *MySQL* – an open-source relational database management system (RDBMS)
<http://www.mysql.com/>