JEQL

A Language for Spatial Processing

(and other things)

Martin Davis

September 2011

Spatial Data Challenges

- **Complex data model**
  - hierarchically structured, arbitrary-length data
  - higher order data structures *(e.g. polygonal coverages)*
- **Numerous (*& complex*) data formats**
  - Shapefile, GML, KML, GeoJSON, GeoRSS, *etc.etc.*
- **Large data volumes**
  - -> Performance is critical
- **Sophisticated algorithms**
  - needed for functionality, performance
  - often supplied by external libs (*e.g. JTS*)
- **Lots of time spent wrangling data**
  - "Spatial ETL"

Lots of Spatial Tools

- **Spatial libraries (JTS)**
  - powerful geometry processing
- **Spatial GUIs (JUMP, etc.)**
  - easy to use
  - graphical display
- **Spatial Databases (PostGIS, etc.)**
  - powerful
  - handles big data
  - SQL good for spatial & attribute manipulation

but...

- **Spatial libraries**
  - complex to use
  - need to be a programmer
- **Spatial GUIs**
  - hard to capture & reuse process chains
  - hard to extend
- **Spatial Databases**
  - locked up inside database
  - hard to access outside resources
  - "impedance mismatch" with external languages

Idea!

- **Higher-level language**
  - easy to use
  - repeatable, sharable, understandable
  - follow SQL paradigm
- **Easy access to external data**
  - data file formats
  - databases
- **Full support for Spatial**
  - datatypes, operations, formats
- **Implement on Java platform**
  - Fast(er) to develop
  - cross-platform
  - excellent tools & libraries

JEQL

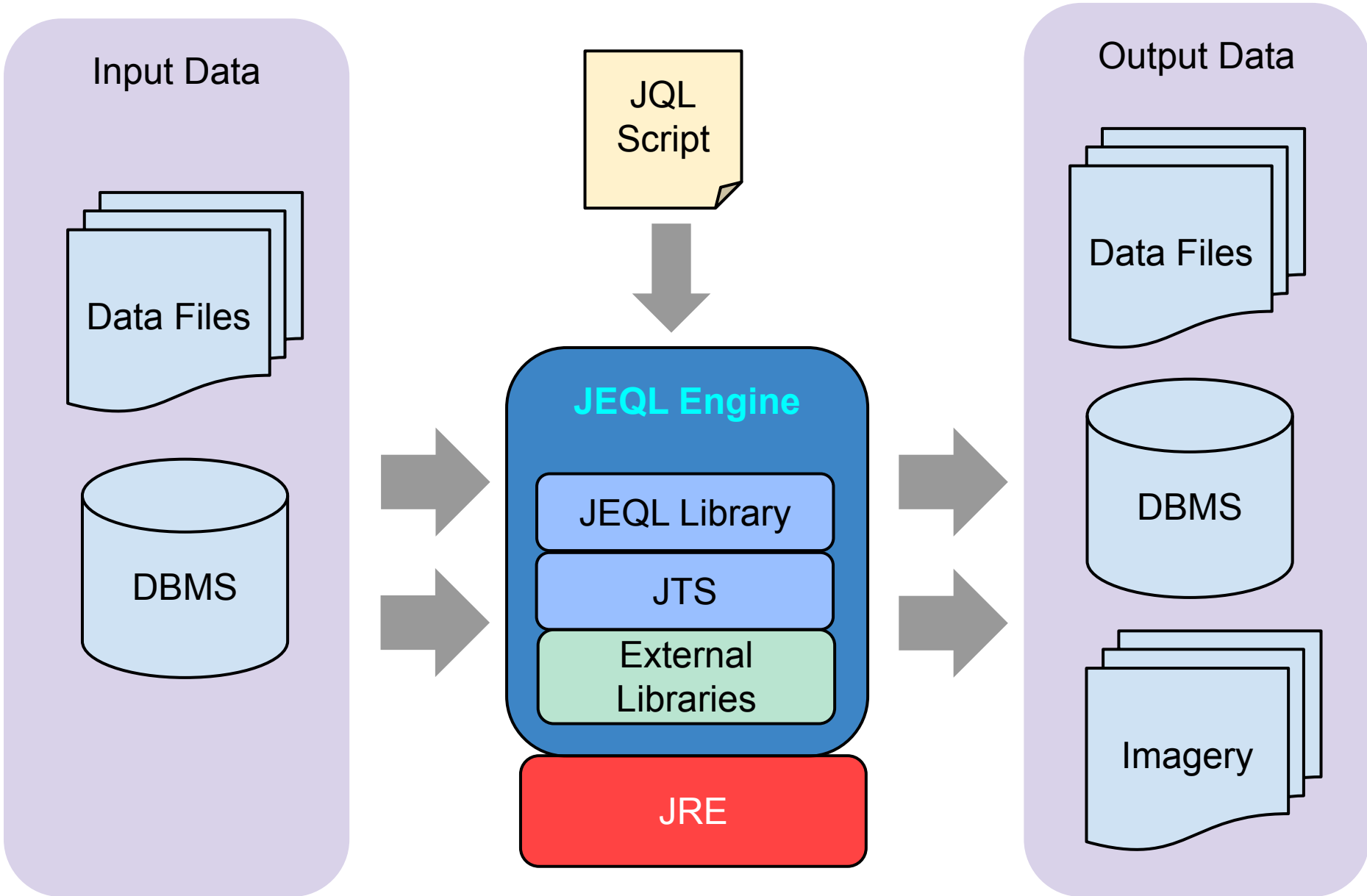
- **Java-based** - runs on the JVM

- **Easy** - simple to learn and use
- **Expressive** - higher-level syntax
- **Efficient** - performant execution model
- **Extensible** - using Java code and libraries
- **Embeddable** - as an engine in other apps
- **ETL** - core use case

- **Query** - SQL-based data manipulation

- **Language**

JEQL Architecture



JEQL Language Features

- **Types**
  - Int, Double, Boolean, String, Date, Geometry
- **Variables, Functions & Expressions**
  - similar to Java
- **Standard Library**
  - String, Geom, RegEx, Date, System, File, etc. etc.
- **Commands**
  - operations which are easier or faster in Java
    - e.g. I/O, complex spatial processes
  - named in/out parameters
  - multiple outputs

JEQL features from Java

- **Interpreted scripting language running on JVM**
  - popular approach - *Clojure, Jython, etc.*

- **Access to Java APIs**
  - Standard Java Library
    - Math, String, RegEx, I/O, JDBC, Java2D, etc.
  - 3rd-party APIs
    - JFreeChart, JTS, Proj4J, GeoTools, etc
- **Performance**
- **Memory management**
- **Easy for developers**
  - IDEs, tools, platform-independent
  - lots of code available
  - simple extensibility (dynamic class loading)

JEQL features from SQL

- **Tables are native datatypes**
  - "Table-Oriented Programming"
  - Tables map 1-1 to databases and other formats
    - low impedance mismatch
- **SELECT expressions**

```
SELECT DISTINCT ... FROM a LEFT OUTER JOIN b
WHERE ... GROUP BY ... ORDER BY ... LIMIT x OFFSET y
```

  - declarative, higher-level code
- **Improvements**
  - use good ideas from others (PostgreSQL!)
  - `WITH` for factoring of common sub-expressions
  - table literals - `TABLE t(a,b,c) ( (...) ... )`
  - `SPLIT BY` - inverse of `GROUP BY`

JEQL features for Spatial

- **Geometry** is first-class data type
  - Values, functions, table columns, etc.
  - WKT Literals - `POLYGON ((1 1, 2 2, ...))`
- **Spatial support provided by JTS**
  - OGC-compliant spatial data model
  - Extensive set of spatial functions
- **Proj4J for Coordinate System transformation**
- **Readers/Writers for spatial formats**
  - File: Shapefile, KML, GML,
  - DB: PostGIS, SQL Server, Oracle, ...
  - easy to add others
- **Spatial indexes** (*future*)

JEQL execution model

- **Streaming Evaluation**
  - Not memory-bound
    - no limit on throughput size
  - No penalty for factoring queries
  - Materialize tables when needed for performance
- **Queries can use row order**
  - different to SQL
  - order is significant in some data formats
  - more powerful processing of serialized datasets

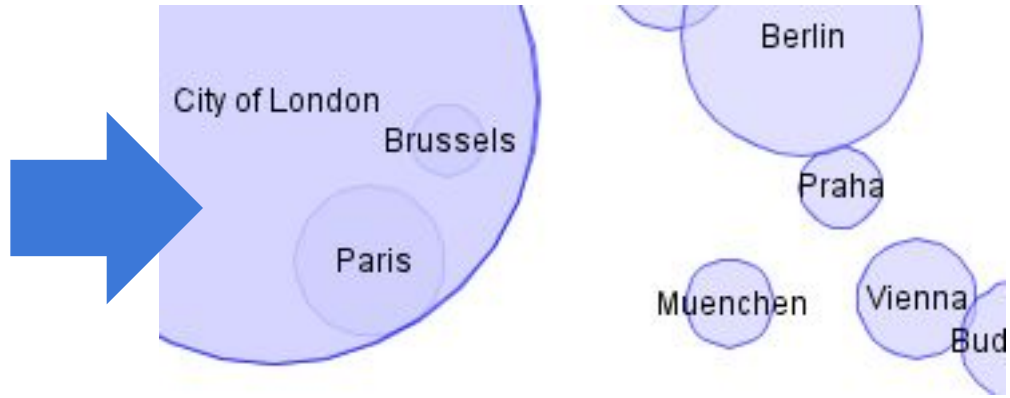
JEQL - Applications

- Data format translation
- Model-to-model transformation
- Data export/import from databases
- Data generation
- Geometry manipulation and processing
- Coordinate system reprojection
- KML generation
- QA and data cleaning
- Plotting
- Charting
- Text processing
- XML/HTML generation

Examples

Read dataset, filter / reproject / buffer, write dataset

```
name,countryCode,lat,lon,pop
"les Escaldes","AD",42.50729,1.53414,15853
"Andorra la Vella","AD",42.50779,1.52109,20430
"Umm al Qaywayn","AE",25.5864,55.57603,44411
"Ra's al Khaymah","AE",25.78953,55.9432,115949
"Khawr Fakkan","AE",25.33132,56.34199,33575
"Dubai","AE",25.25817,55.30472,1137347
"Diba","AE",25.61955,56.27291,26395
"Sharjah","AE",25.35731,55.4033,543733
"Ar Ruways","AE",24.11028,52.73056,16000
```




```
CSVReader tcityRaw hasColNames: file: "cities.csv";

//-- convert raw CSV to typed values
tcity = select name,
                Val.toDouble(lat) lat,
                Val.toDouble(lon) lon,
                Val.toInt(pop) population
        from tcityRaw;

//--- PROJ4-style projections
projGoogleMerc = "+proj=merc +a=6378137 +b=6378137 +lat_ts=0.0 +lon_0=0.0 +x_0=0.0 +y_

//--- select large cities, buffer by population
tcityBuf = select name,
                population,
                Geom.buffer(CRS.project(Geom.createPoint(lon, lat), projGoogleMerc),
                        population / 10) cityBuf
        from tcity where population >= 1_000_000;

ShapefileWriter tcityBuf file: "cities_proj.shp";
```

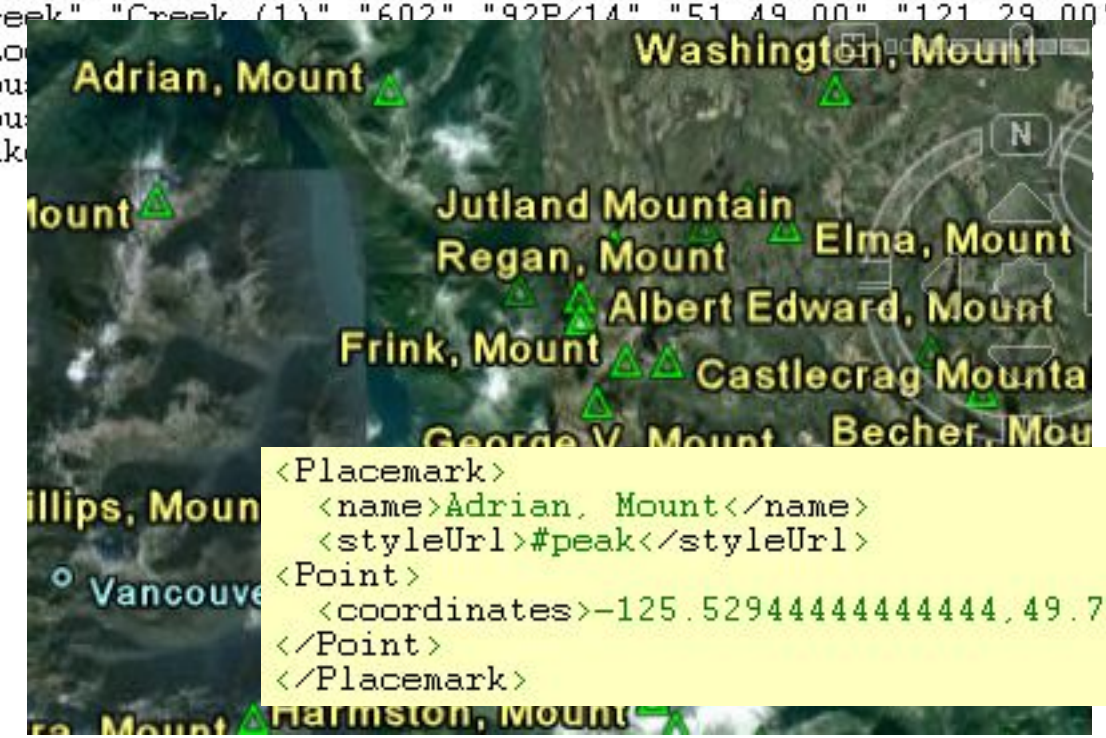
Generate GML

```
//--- Data provided in table literal
t = TABLE t(id, type, lanes, Geometry) (
( 1, "HWY",  4, LINESTRING (1 1, 2 2))
( 2, "CITY", 2, LINESTRING (10 10, 20 20))
( 3, "CITY", 2, LINESTRING (100 100, 20 20, 300 300, 400 400))
) ;

//--- define GML id
tgml = select id gml_id, * from t;

GMLWriter tgml
        namespacePrefix: "ts"
        namespaceURI: "http://tsusiatsoftware.net/gml"
        featureCollectionTag: "RoadNetwork"
        featureTag: "RoadSegment"
        file: "test.gml";
```

```
▼<RoadNetwork xmlns:gml="http://www.opengis.net/gml" xmlns:ts="http://tsusiatsoftware
  ▼<gml:featureMember>
    ▼<RoadSegment fid="1">
       <ts:id>1</ts:id>
       <ts:type>HWY</ts:type>
       <ts:lanes>4</ts:lanes>
      ▼<ts:Geometry>
        ▼<gml:LineString>
           <gml:coordinates>1.0,1.0 2.0,2.0</gml:coordinates>
         </gml:LineString>
       </ts:Geometry>
     </RoadSegment>
   </gml:featureMember>
  ▼<gml:featureMember>
    ▼<RoadSegment fid="2">
       <ts:id>2</ts:id>
```

Convert CSV File to KML - input & output

```
"Official Name","Feature Type","Feature Type Code","Mapsheet","Latitude","Longitude"
"'Adade Yus Mountain","Mountain","2701","93N/7","55 18 00","124 41 00"
"'Ksan","Locality","108","93M/4","55 15 00","127 40 00"
"100 Mile House","District Municipality (1)","35","92P/11","51 39 00","121 17 00"
"101 Mile Lake","Lake","951","92P/11","51 40 00","121 18 00"
"103 Mile Lake","Lake","951","92P/11","51 41 00","121 18 00"
"105 Mile House","Locality","108","92P/11","51 42 00","121 19 00"
"105 Mile Lake","Lake","951","92P/11","51 43 00","121 20 00"
"105 Mile Post 2, Réserve indienne","Réserve indienne","552","92I/11","50 44 00","121 19
"105 Mile Post Indian Reserve 2","Indian Reserve","543","92I/11","50 44 00","121 19 00"
"108 Mile Lake","Lake","951","92P/11","51 45 00","121 21 00"
"108 Mile Ranch","Community","121","92P/11","51 45 00","121 21 00"
"111 Mile Creek","Creek (1)","602","92P/14","51 48 00","121 28 00"
"111 Mile House","Locality","108","92P/14","51 46 00","121 23 00"
"114 Mile House","Locality","108","92P/14","51 48 00","121 26 00"
"117 Mile Creek","Creek (1)","602","92P/14","51 49 00","121 29 00"
"12 Mile","Lo
"122 Mile Hou
"127 Mile Hou
"130 Mile Lak
```

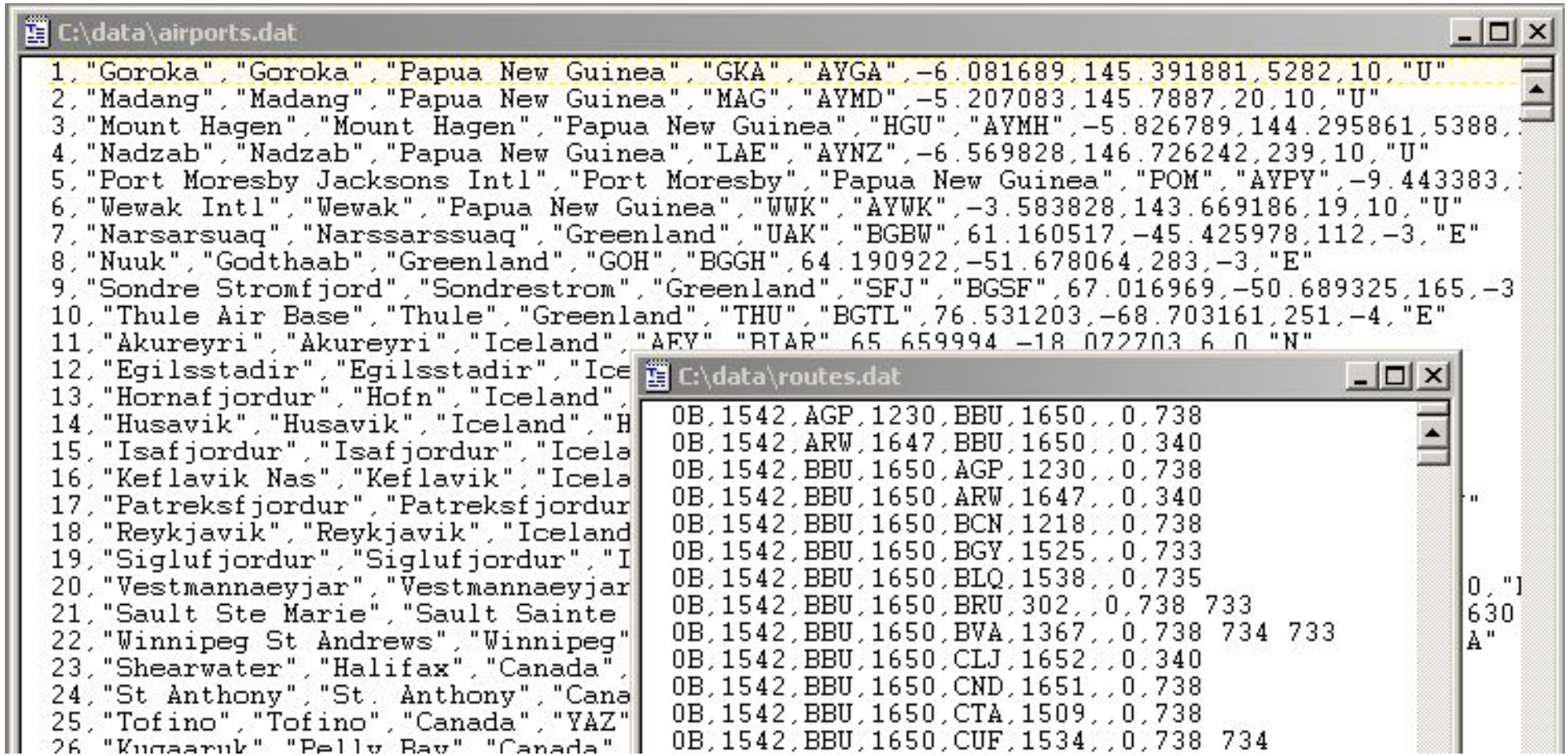



```
<Placemark>
  <name>Adrian, Mount</name>
  <styleUrl>#peak</styleUrl>
<Point>
  <coordinates>-125.52944444444444,49.756388888888885</coordinates>
</Point>
</Placemark>
```

Convert CSV File to KML - script

```
//--- Read data from CSV
CSVReader tCSV useColNames: file: "BC GAZETTEER 8Oct07.csv";

//--- Select mountain features
tMt = select * from tCSV where String.leftStr(Feature_Type, 5) == "Mount";

//--- Prepare KML dataset with name, location as POINT, named style
tKML = select    nameSafe as kmlName,
                 Geom.createPoint(lon, lat) as geometry,
                 "#peak" as kmlStyleUrl
        with {
        nameSafe = String.replaceAll(Official_Name, "&", "&amp;");
        //--- convert textual DDMMSS to decimal degrees
        lat =    Val.toDouble(String.substring(Latitude, 0, 2))
                 + Val.toDouble(String.substring(Latitude, 3, 5)) / 60.0
                 + Val.toDouble(String.substring(Latitude, 6, 8)) / 3600.0;
        lon =    - (Val.toDouble(String.substring(Longitude, 0, 3))
                 + Val.toDouble(String.substring(Longitude, 4, 6)) / 60.0
                 + Val.toDouble(String.substring(Longitude, 7, 9)) / 3600.0);
        }
        from tMt;

//--- KML styling for named style
tStyle = select * from table (
  ("peak", "#ff7fffff", "#ff00ff00", 0.6,
                        "http://maps.google.com/mapfiles/kml/shapes/triangle.png")
        ) styles(id, labelStyleColor, iconStyleColor, iconStyleScale, iconStyleHref);

KMLWriter tKML
        styles: tStyle
        name: "B.C. Placenames"
        file: "bcNames.kml";
```

Air Routes Visualization - Data



Air Routes Visualization - prepare data

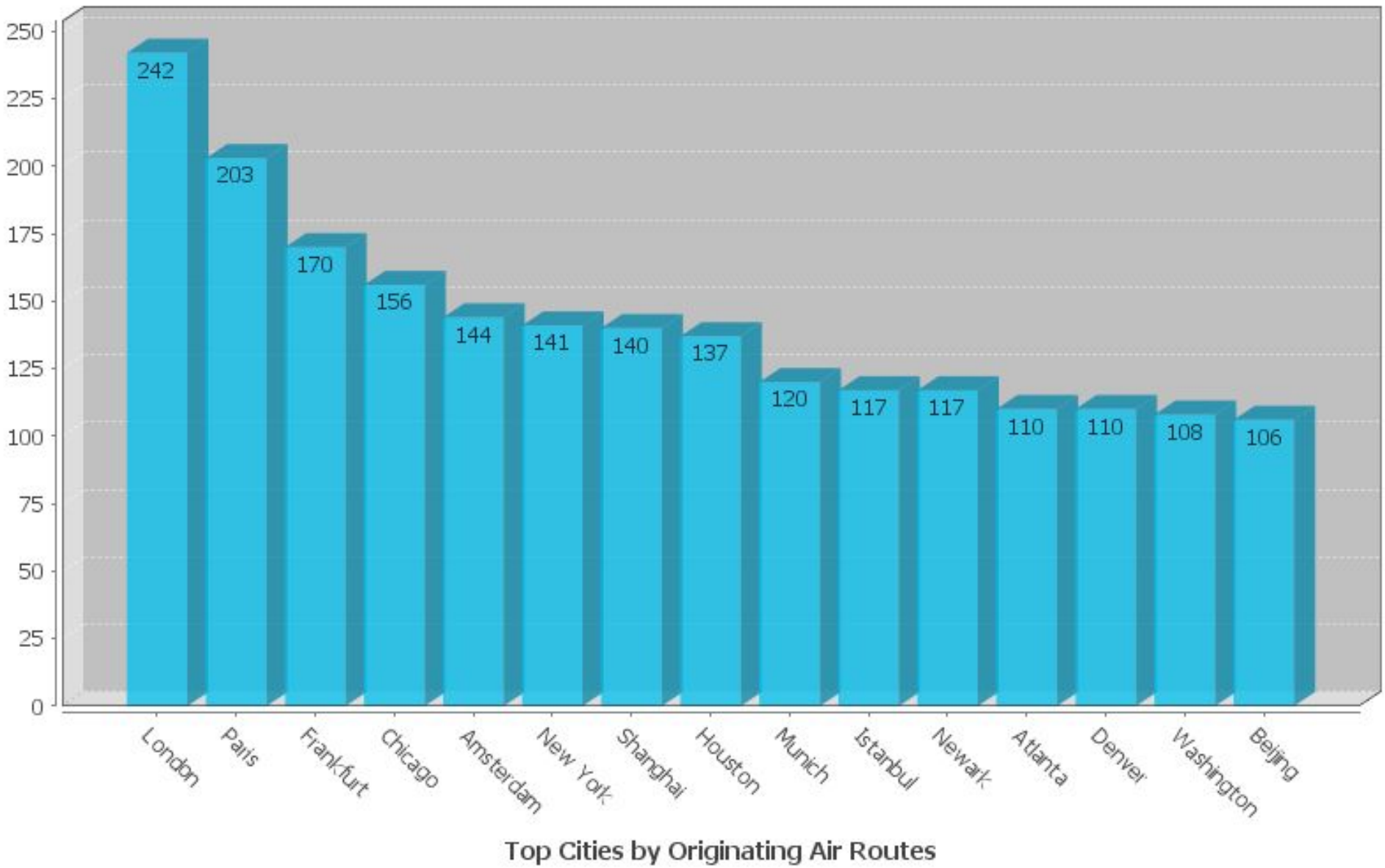
```
// Read airports with Id, city, and location
CSVReader tairportRaw file: "airports.dat";
tairport = select Val.toInt(col1) id,
                  col3 city,
                  col4 ctry,
                  Val.toDouble(col7, 0.0) lat,
                  Val.toDouble(col8, 0.0) lon
                  from tairportRaw;

// Read routes with from and to city IDs
CSVReader trouteRaw file: "routes.dat";
trouteAll = select rownum() rteID,
                  Val.toInt(col4, -1) fromID,
                  Val.toInt(col6, -1) toID
        from trouteRaw;

// Keep only a single route for each city pair
troute = select distinct fromID, toID from trouteAll;

//-- compute route vectors by joining routes & airports
troute2 = select
        fromID, a1.city fromCity, a1.ctry fromCtry, a1.lat fromLat, a1.lon fromLon,
        toID,   a2.city toCity,   a2.ctry toCtry,   a2.lat toLat,   a2.lon toLon
        from troute r
        join tairport a1 on r.fromID == a1.id
        join tairport a2 on r.toID   == a2.id;

CSVWriter troute2 colNames: file: "routeLine.csv";
```

Air Routes Visualization -Plot

Air Routes Visualization - plot geodetic arcs

```
//----- Plot a global map of Air Routes
CSVReader troute hasColNames: file: "routeLine.csv";

trte = select   fromCity, toCity,
                Val.toDouble(fromLon) fromLon, Val.toDouble(fromLat) fromLat,
                Val.toDouble(toLon)   toLon,   Val.toDouble(toLat)   toLat
       from troute;

//----- Generate geodetic arcs
tlines = select fromCity, toCity, line, Geom.length(line) len
       with {
         line = Geodetic.split180(Geodetic.arc(fromLon, fromLat, toLon, toLat, 2));
       }
       from trte order by len desc;

//----- Colour-theme arcs by length
tplot = select line,
       Color.interpolate("ffffff", "0000ff", "000080", len / 80.0 ) lineColor,
       0.4 lineWidth
       from tlines;

//----- Plot world country polygons and borders for context
ShapefileReader tworld file: "world.shp";
tworldLine = select GEOMETRY, "22222277" lineColor from tworld;
tworldFill = select GEOMETRY, "333333"   fillColor from tworld;

width = 4000;
Plot   width: width    height: width / 2
       extent: LINESTRING(-180 -90, 180 90)
       data: tworldFill
       data: tplot
       data: tworldLine
       file: "routes.png";
```

Charting - output



Charting - script

```
CSVReader troute hasColNames: file: "routeLine.csv";

//--- Compute counts of routes from each city
tCityCount = select count(*) cnt, fromCity city
                from troute
                group by fromCity
                order by cnt desc;

//--- Limit to top 15
tchart = select city key, cnt value from tCityCount limit 15;

//--- Generate 3D bar chart with labels, axes
Chart type: "bar"
        data: tchart
        extrude:
        color: "00c0f0"
        showItemLabels:
        xAxisTitle: "Top Cities by Originating Air Routes"
        xAxisLabelRotation: -0.5
        width: 800
        file: "cityRoutes.png";
```

Database access

```
//--- read data from database query
DbReader t
        driver: "org.h2.Driver"
        url: "jdbc:h2:tcp://localhost/~/test"
        user: "sa"
        password: ""
        sql: $"
select f.id, f.name, loc.x, loc.y
from feature f
join location loc on f.id = loc.id
order by id" ;

//--- process data
t2 = select id, name, Geom.createPoint(x, y) from t;

//--- write data to DB table
DbWriter t2
        table: "TEST"
        driver: "oracle.jdbc.driver.OracleDriver"
        url: "jdbc:oracle:thin:@localhost:1521:xe"
        user: "test"
        password: "test"
```

JEQL for Developers

- **Easy to extend**
  - Functions are Java `public static` methods
  - Commands are Java classes
- **Easy to develop & debug**
  - Run & debug in any Java IDE
- **JEQL Engine is embeddable in other applications**
  - expose namespace of tables to JEQL
  - register functions & commands
  - e.g. Use as a query engine for OpenJUMP

Comparison with other products

- **ETL Tools**
  - OGR
  - GeoKettle, etc.
- **Spatial Processing Tools**
  - GeoScript
  - GMT - Generic Mapping Tools
- **Spatial Databases**
  - PostGIS, etc.
  - Spatialite, H2 (Jaspa, GeoDB), etc.

Future Plans

- **More data formats**
- **Better integration with databases**
  - Express DB queries directly in JEQL SQL
- **Spatial Indexing**
  - Automatically created when required
- **GUI Builder for scripts**
- **JEQL Workbench**
  - GUI for developing/running scripts
  - Viewer for tabular and spatial datasets
  - Visual programming?
  - *High-performance Integrated Development Environment*

Distribution & Support

- **Engine is freely available for download & distribution**
  - Open Source soon
- **Documentation**
  - Reference
  - Examples
- **Mailing List**

http://tsusiatsoftware.net/jeql/main.html