Navigating the spatial data support in MySQL 4.1

Navigating-The-Spatial-Data-Support.sxi

Lenz Grimmer <lenz@mysql.com>
Senior Release Engineer

International PHP Conference 2003
Frankfurt , Germany

Content of this session

- What geometries are
- OpenGIS geometry classes and their properties
- Geometry data exchange formats
- Creating a spatial database in MySQL
- Analysing spatial information
- Optimizing analysis by use of spatial indexes
- World map application demo

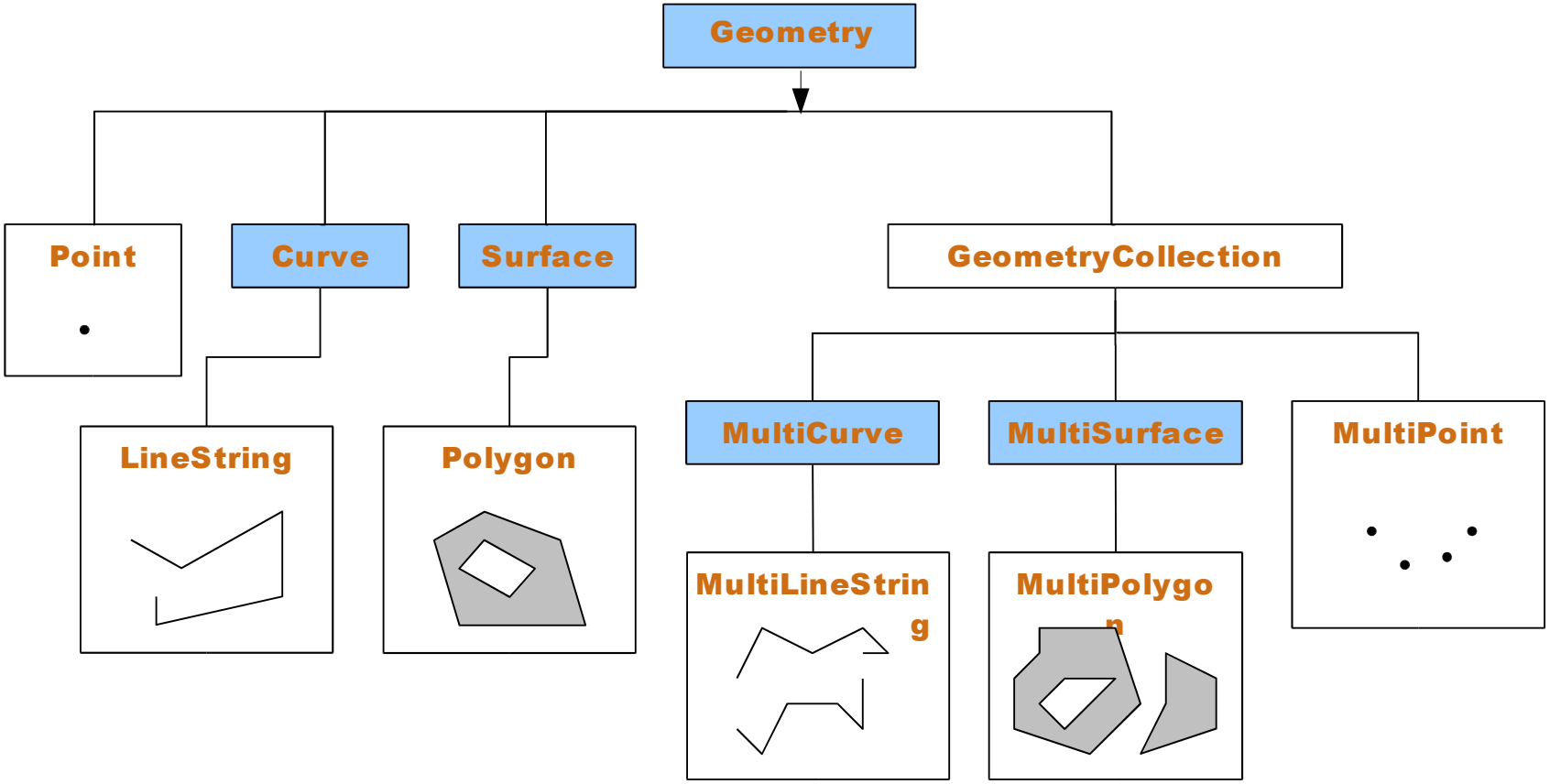
OGC and OGC specifications

- MySQL implements spatial extensions following the specification of the OpenGIS Consortium (OGC)
  http://www.opengis.org/
- OpenGIS Simple Feature Specifications For SQL
  http://www.opengis.org/techno/implementation.l
- MySQL implements a subset of the **SQL With Geometry Types environment**
- The specifications describe a set of SQL geometry types, as well as functions on those types to create and analyse geometry values.

Term definitions

- A *geographic/geospatial feature* is anything in the world that has a location.
- Feature examples:
  - An entity. For example, a mountain, a pond, a city.
  - A space. For example, a postcode area, the tropics.
  - A definable location. For example, a crossroad, as a particular place where two streets intersect.
- *Geometry* is another word that denotes a geographic feature.
- General definition: “A point or an aggregate of points representing anything in the world that has a location.”

OpenGIS Geometry Model



Class Geometry

A Geometry value has the following properties:

- The **type** that a geometry belongs to.
- Its **SRID**, the identifier of a geometry's associated Spatial Reference System.
- Geometry's **coordinates** (related to the SRID).
- Its **interior, boundary and exterior**.
- Its **MBR** (Minimum bounding rectangle), or **Envelope**: ((MINX MINY, MAXX MINY, MAXX MAXY, MINX MAXY, MINX MINY))
- The quality of being **simple or non-simple**.
- The quality of being **closed or not closed**.
- The quality of being **empty or not empty**.
- Its **dimension**:
  - -1 == empty geometries.
  - 0 == no length and no area. Points and MultiPoints.
  - 1 == non-zero length and no area. LineStrings and MultiLineStrings.
  - 2 == non-zero area. Polygons and MultiPolygons.

Class Point

- A Point is a geometry that represents a single location in coordinate space.
- Examples:
  - A city on a world map.
  - A bus stop on a city map.
- Point properties:
  - X-coordinate value.
  - Y-coordinate value.
  - The Boundary of a Point is an empty set.
  - The dimension is 0.

Class Curve

- A Curve is an one-dimensional geometry, usually represented by a sequence of points
- Curve properties:
  - Coordinates of its points.
  - Dimension is 1.
  - A Curve is closed if its start point is equal to its end point.
  - The boundary of a closed Curve is empty.
  - The boundary of a non-closed Curve consists of its two end points.
  - A Curve is simple if it does not pass through the same point twice.
  - A simple closed Curve is a Ring.

Class LineString

- A LineString is a Curve with linear interpolation between points.
- LineString examples:
  - A river on a world map.
  - A street on a city map.

LineString properties:

- Coordinates of LineString segments.
- A LineString consisting of two points is a Line.
- A closed simple LineString is a LinearRing.

Class Surface

- A Surface is a two-dimensional geometry. It is a non-instantiable class. Its only instantiable subclass is Polygon.
- Surface properties:
  - Dimension is 2.
  - OpenGIS specification defines a simple Surface as a geometry that consists of a single "patch" that is associated with a single exterior boundary and zero or more interior boundaries.
  - The boundary is a set of closed curves.

Class Polygon

- A Polygon is a planar Surface representing a multisided geometry
- Polygon examples:
  - A park on a city map
  - A country on a world map
- The assertions for polygons:
  - The boundary of a Polygon is a set of LinearRings.
  - No two rings in the boundary cross. No cut lines, spikes or punctures.
  - Polygons are simple geometries

Class GeometryCollection

- A GeometryCollection is a collection of one or more geometries of any class.
- Subclasses restrictions:
  - All Elements must be in the same Spatial Reference System
  - All Elements must be of the same type.
  - Dimension.
  - Other constraints (e.g. on the degree of spatial overlap between elements).

Class MultiPoint

- A MultiPoint is a collection of Points.
- MultiPoint examples:
  - An archipelago of islands on a world map.
  - A set of a sport center's ticket windows on a city map.
- MultiPoint properties:
  - Dimension is 0.
  - A MultiPoint is simple if no two Points are equal.
  - The boundary is empty set.

Class MultiCurve

- A MultiCurve is a collection of Curves.
- MultiCurve properties:
  - A MultiCurve is simple if elements are simple and don't intersect.
  - A MultiCurve is defined as a one-dimensional geometry.
  - A MultiCurve is closed if all of its elements are closed.
  - "Mod 2 union" rule for the boundary (A point is in the boundary of a MultiCurve if it is in the boundaries of an odd number of MultiCurve elements).

Class MultiLineString

- A MultiLineString is a MultiCurve whose elements are LineStrings.

- MultiLineString examples:
  - A river with its tributaries on a world map.
  - A highway system on a city map.

Class MultiSurface

A MultiSurface is a collection of surfaces.

- MultiSurface assertions:
  - The interiors may not intersect.
  - The boundaries may intersect at a finite number of points.

Class MultiPolygon

- MultiPolygon is a MultiSurface whose elements are Polygons.
- MultiPolygon examples:
  - A lakes system on a world map.
  - A set of towers of the same building on a city map.
- MultiPolygon assertions:
  - The interiors of two Polygons that are elements of a MultiPolygon may not intersect.
  - The boundaries of elements may not cross
  - The boundaries of elements may touch a finite number of points.
- MultiPolygon properties:
  - MultiPolygon is defined as two-dimensional geometry.
  - The boundary of a MultiPolygon is a set of closed LineStrings corresponding to the boundaries of its elements.

Supported Spatial Data Formats

- Well-Known Text (WKT) representation
  - Designed to exchange geometry data in ASCII format
- Well-Known Binary (WKB) representation
  - Designed to exchange geometry data as binary streams
  - Defined in the OpenGIS specifications as well as in the ISO "SQL/MM Part 3: Spatial" standard.
- Internally, MySQL stores geometry values in a format that is not identical to either WKT or WKB format.
- Currently, only MyISAM tables fully support spatial data columns with indexes.

Well-Known Text (WKT) Representation

- WKT examples:
  - `POINT(10 10)`
  - `LINESTRING(10 10, 20 20, 30 40)`
  - `POLYGON((10 10, 10 20, 20 20, 20 15, 10 10))`
  - `MULTIPOINT(10 10, 20 20)`
  - `MULTILINESTRING((10 10, 20 20), (15 15, 30 15))`
  - `MULTIPOLYGON(((10 10, 10 20, 20 20, 20 15, 10 10)), ((60 60, 70 7, 80 60, 60 60 )))`
  - `GEOMETRYCOLLECTION(POINT(10 10), POINT(30 30), LINESTRING(15 15, 20 20))`
- BNF of WKT: http://www.opengis.org/techno/implementation.htm

Well-Known Binary (WKB) Representation

`POINT(1,1)` in WKB representation:

`0101000000000000000000F03F000000000000F03F`

Byte order : 01
WKB type : 01000000
X : 000000000000F03F
Y : 000000000000F03F

MySQL Spatial Data Types

- GEOMETRY
- POINT
- LINESTRING
- POLYGON
- MULTIPOINT
- MULTILINESTRING
- MULTIPOLYGON
- GEOMETRYCOLLECTION

Functions to create a geometry using its WKT

- `GeomFromText(wkt,srid)`
- `PointFromText(wkt,srid)`
- `LineFromText(wkt,srid)`
- `PolyFromText(wkt,srid)`
- `MPointFromText(wkt,srid)`
- `MLineFromText(wkt,srid)`
- `MPolyFromText(wkt,srid)`
- `GeomCollFromText(wkt,srid)`

Functions to create a geometry using its WKB

- `GeomFromWKB(wkt,srid)`
- `PointFromWKB(wkt,srid)`
- `LineStringFromWKB(wkt,srid)`
- `PolyFromWKB(wkt,srid)`
- `MPointFromWKB(wkt,srid)`
- `MLineFromWKB(wkt,srid)`
- `MPolyFromWKB(wkt,srid)`
- `GeomCollFromWKB(wkt,srid)`

Creating Spatial Columns

CREATE TABLE

```
mysql> CREATE TABLE g1 (p1 GEOMETRY);
Query OK, 0 rows affected (0.02 sec)
```

ALTER TABLE

```
mysql> ALTER TABLE g1 ADD p2 POINT;
Query OK, 0 rows affected (0.00 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

Populating Spatial Columns

Using WKT functions:

```
INSERT INTO geom VALUES (GeomFromText('POINT(1 1)'))
INSERT INTO geom VALUES (GeomFromText('POLYGON((0 0,10 0,10
   10,0 10,0 0),(5 5,7 5,7 7,5 7, 5 5))'))
INSERT INTO geom VALUES (GeomFromText('GEOMETRYCOLLECTION
   (POINT(1 1),LINESTRING(0 0,1 1,2 2,3 3,4 4))'))
INSERT INTO geom VALUES (PointFromText('POINT(1 1)'))
INSERT INTO geom VALUES (LineStringFromText('LINESTRING(0 0,1
   1,2 2)'))
INSERT INTO geom VALUES (GeomCollFromText('GEOMETRYCOLLECTION
   (POINT(1 1),LINESTRING(0 0,1 1,2 2,3 3,4 4))'))
```

Using WKB functions:

```
INSERT INTO geom VALUES (GeomFromWKB
   (0x0101000000000000000000F03F000000000000F03F));
INSERT INTO geom VALUES (GeomFromWKB(?));
INSERT INTO geom VALUES (GeomFromWKB
   ('\0\0\0\0\0\0\0\0p?\0\0\0\0\0\0p?'));
```

Fetching Spatial Data

Using WKT Representation:

```
mysql> select AsText(p1) from g1;
+--------------------------+
| AsText(p1)               |
+--------------------------+
| POINT(1 1)               |
| LINESTRING(0 0,1 1,2 2)  |
+--------------------------+
2 rows in set (0.00 sec)
mysql>
```

Using WKB Representation:

```
SELECT AsBinary(g) FROM geom;
```

Analysing Spatial Data

Ways of Analysing Spatial Data.

- Interactive SQL program: mysql or MySQLCC.
- Application programs in any languages supporting a MySQL client API.

Four major function groups:

- Functions that convert geometries between various formats.
- Functions that describe qualitative or quantitative properties of a geometry.
- Functions that describe relations between two geometries.
- Functions that create new geometries from existing ones.

Functions To Convert Between Geometries Between Different Formats

GeomFromWKT(string wkt [,integer srid]): geometry

- Converts WKT representation into internal geometry format.

GeomFromWKB(binary wkb [,integer srid]): geometry

- Converts WKB representation into internal geometry format

AsWKT(geometry g): string

- Converts internal geometry format into WKT representation.

AsWKB(geometry g): binary

- Converts internal geometry format into WKB representation.

```
mysql> select AsText(GeomFromText('LineString(1 1,2 2,3
   3)'));
+-------------------------------------------------+
| AsText(GeomFromText('LineString(1 1,2 2,3 3)')) |
+-------------------------------------------------+
| LINESTRING(1 1,2 2,3 3)                         |
+-------------------------------------------------+
```

Functions To Analyse Geometry Properties

```
mysql> select GeometryType(GeomFromText('POINT(1 1)'));
+-----------------------------------------+
| GeometryType(GeomFromText('POINT(1 1)')) |
+-----------------------------------------+
| POINT                                   |
+-----------------------------------------+

mysql> select Dimension(GeomFromText('LineString(1 1,2 2)'));
+-----------------------------------------------+
| Dimension(GeomFromText('LineString(1 1,2 2)')) |
+-----------------------------------------------+
| 1                                             |
+-----------------------------------------------+

mysql> select SRID(GeomFromText('LineString(1 1,2 2)',101));
+-----------------------------------------------+
| SRID(GeomFromText('LineString(1 1,2 2)',101)) |
+-----------------------------------------------+
| 101                                           |
+-----------------------------------------------+

mysql> select AsText(Envelope(GeomFromText('LineString(1 1,2 2)',101)));
+-------------------------------------------------------------+
| AsText(Envelope(GeomFromText('LineString(1 1,2 2)',101))) |
+-------------------------------------------------------------+
| POLYGON((1 1,2 1,2 2,1 2,1 1))                              |
+-------------------------------------------------------------+
```

Functions To Analyse Point Properties

`X(point p):Double`

- The x-coordinate value for this point.

`Y(point p):Double`

- The y-coordinate value for this point

Functions To Analyse LineString Properties

`StartPoint(LineString l):Point`

- The start point of this LineString.

`EndPoint(LineString l):Point`

- The end point of this LineString.

`PointN(LineString l,integer n):Point`

- Returns the specified point N in this Linestring.

`GLength(LineString l):Double`

- The length of this LineString in its associated spatial reference.

`IsClosed(LineString l):Integer`

- Returns 1 (TRUE) if this LineString is closed (StartPoint() == EndPoint()).

`NumPoints(LineString l):Integer`

- The number of points in this LineString.

Functions To Analyse MultiLineString Properties

`GLength(MultiLineString m):Double`

- The Length of this MultiLineString which is equal to the sum of the lengths of the elements.

`IsClosed(MultiLineString m):Integer`

- Returns 1 (TRUE) if this MultiLineString is closed (StartPoint() = EndPoint() for each LineString in this MultiLineString).

Functions To Analyse Polygon Properties

`Area(Polygon p):Double`

- The area of this Polygon, as measured in the spatial reference system of this Polygon.

`NumInteriorRings(Polygon p):Integer`

- Returns the number of interior rings in this Polygon.

`ExteriorRing(Polygon p):LineString`

- Returns the exterior ring of this Polygon as a LineString.

`InteriorRingN(Polygon p, Integer N): LineString`

- Returns the N-th interior ring for this Polygon as a LineString.

Functions To Analyse MultiPolygon Properties

`Area(MultiPolygon m):Double`

- The area of this MultiPolygon, as measured in the spatial reference system of this MultiPolygon.

Functions To Analyse GeometryCollection Properties

`NumGeometries(GeometryCollection g):Integer`

- Returns the number of geometries in this GeometryCollection.

`GeometryN(GeometryCollection g,integer N): Geometry`

- Returns the N-th geometry in this GeometryCollection.

TODO: Functions That Create New Geometries From Existing Ones

Spatial Operators

```
Intersection(Geometry g1,g2):Geometry
Union(Geometry g1,g2):Geometry
Difference(Geometry g1,g2):Geometry
SymDifference(Geometry g1,g2):Geometry
Buffer(Geometry g, double d):Geometry
ConvexHull(Geometry g):Geometry
```

Testing Relations On Geometry Minimal Bounding Rectangles

```
MBRContains(geom1,geom2)
MBRWithin(geom1,geom2)
MBRDisjoint(geom1,geom2)
MBREqual(geom1,geom2)
MBRIntersects(geom1,geom2)
MBROverlaps(geom1,geom2)
MBRTouches(geom1,geom2)
```

TODO: Functions That Test Spatial Relationships Between Geometries

```
Contains(geom1,geom2)
Crosses(geom1,geom2)
Disjoint(geom1,geom2)
Equal(geom1,geom2)
Intersects(geom1,geom2)
Overlaps(geom1,geom2)
Touches(geom1,geom2)
Within(geom1,geom2)
```

Spatial Indexes

The most typical database searches:

- Searching for all values that are greater than a given one.
- Searching for all values that are less than a given one.
- Searching for all values that are equal to a given one.

The most typical spatial searches:

- A point query. Searching for all objects that contain a given point.
- A region query. Searching for all objects that overlap a given region.

MySQL utilizes R-Trees with quadratic splitting to index spatial columns.

Creating Spatial Indexes

- Creating with ALTER TABLE:

```
ALTER TABLE g ADD SPATIAL KEY(g);
```

- With CREATE TABLE:

```
CREATE TABLE g (g GEOMETRY NOT NULL,
SPATIAL INDEX(g));
```

- With CREATE INDEX:

```
CREATE SPATIAL INDEX sp_index ON g
(g);
```

Using a Spatial Index

Spatial search without index

```
mysql> select fid,AsText(g) from g where
mysql> MBRContains(GeomFromText('Polygon((30000 15000,31000 15000,31000 16000,30000
    16000,30000 15000))'),g);
.....
20 rows in set (0.46 sec)
```

Creating an index:

```
mysql> ALTER TABLE g ADD SPATIAL KEY(g);
Query OK, 32376 rows affected (4.05 sec)
Records: 32376 Duplicates: 0 Warnings: 0
```

The same search with index

```
mysql> SELECT fid,AsText(g) FROM g WHERE
mysql> MBRContains(GeomFromText('Polygon((30000 15000,31000 15000,31000 16000,30000
    16000,30000 15000))'),g);
.....
20 rows in set (0.00 sec)
```

Examining query plan with EXPLAIN

```
mysql> EXPLAIN SELECT fid,AsText(g) FROM g WHERE
mysql> MBRContains(GeomFromText('Polygon((30000 15000,31000 15000,31000 16000,30000 16000,30000 15000))'),
    g);
+----+-------------+-------+-------+---------------+------+---------+------+------+-------------+
| id | select_type | table | type  | possible_keys | key  | key_len | ref  | rows | Extra       |
+----+-------------+-------+-------+---------------+------+---------+------+------+-------------+
| 1 | SIMPLE | g | range | g | g | 32 | NULL | 50 | Using where |
+----+-------------+-------+-------+---------------+------+---------+------+------+-------------+
1 row in set (0.00 sec)
```

OpenGIS Features That Are Not Yet Implemented

- `Union(geom1,geom2)` - spatial operators.
- `Contains(geom1,geom2)` - non-MBR, accurate spatial relations.
- `Distance(geom1,geom2)` - distance between two geometries.
- `Area(GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',SRID))` - should return a result depending on SRID.

GIS application demo

- A small C application that uses the MySQL client library and the GD library to draw a zoomable world map
- It can be used either in standalone mode or as a CGI script
- All map data is stored inside a MySQL database
- SQL queries extract the data relevant for the requested coordinates
- Source code is available online at

http://www.mysql.com/Downloads/Presentations/MySQL-User-Conference-2003/gd.tar.gz