**CSE 510 – Project Phase 2**

**Spatial Data in Java minibase**

**Project Members-**

Patrick Gaines- 1205565669

Omkar Totade- 1209345380

Robert Stevens- 1001151667

Sitanshu Mishra- 1206136577

**Project Phase 2 Objective-**

Our aim for the project was to understand the Java minibase at the Logical Level. We had to understand what type of data is inserted, processed and stored, how the SQL queries are parsed and processed, and finally how the data is retrieved at the logical level.

Finally, we had to modify the minibase so that it could process Spatial Data (similar to the type SDO\_GEOM specified by Oracle) and allow it to create tables to store the spatial data, insert spatial data into them and then perform certain selection queries on it for calculating the area of a stored shape, distance between two shapes and lastly check for the intersection of two given shapes.

**Our implementation-**

Our implementation supports 3 shapes, viz. rectangle, triangle and circle.

We had to make changes to the attribute types that the minibase was using, to allow it to include spatial data of the type Sdo\_Geom and an array of type double to include the coordinates associated with every shape. We declared a separate class called SDOGeometry that would have an enumeration of the shapes that our code supports and also the area (), distance () and intersection () functions that we would require to calculate each of those later.

Subsequently we also had to modify the Convert.java with a few getters and setters so that it would also convert the new SDOGeom data type to a byte array. Similarly, the files TupleUtils.java and Projection.java also required a few minor changes to support the new data type. Finally, we had to modify the Tuple.java file also since a new tuple would have to include data of the type SDOGeom and a double array.

To make different tests for creating, inserting, indexing and selections, we looked at the already existing tests that we were provided with and worked our way around it to include the new functionality.

**Files Changed/Added-**

global -> AttrType.java

global -> Convert.java

global -> SDOGeometry.java

heap -> Tuple.java

iterator -> TupleUtils.java

iterator -> Projection.java

tests -> Test1.java

tests -> Test2.java

tests -> Test3.java

tests -> Test4.java

tests -> Test5.java

tests -> Test6.java

**Test Files-**

We wrote a total six test files for testing out each of the new functionality. A list of what each test file does is as below,

**Test1-** Create a ShapesTable (stores ShapeId, ShapeName and SDOGeometry that stores a shape and its coordinates). Also, create a SDOGeoMetadata table that needs to be created every time we use SDOGeometry (according to Oracle’s tutorial on SDO\_GEOM package).

**Test2-** Insert rows into the ShapesTable and the SDOGeoMetadata tables.

**Test3-** Creates an index on ‘shapeName’ attribute of the ShapesTable. We are using the already existing BinaryTreeFile to create an index on a String field.

**Test4-** Calculates distance between two shapes using a select query. The ‘shapeName’ attribute is used to identify the shapes from the ShapesTable

**Test5-** Calculates area of a shape in the table using a select query. The shapeName attribute is used to retrieve the shape and calculate its area.

**Test6-** Calculates the intersection of any two given shapes in the database. It uses the ‘shapeName’ attribute to identify the shapes from the table and then calculates the points of intersection.

All the queries are hardcoded in their respective test cases and can be changed by hard coding different attributes. More rows can be inserted, different shapes can be used to calculate area, distance and intersection.

**Area (), Intersection (), Distance ()-**

We have implemented the area (), intersection () and the distance () functions in the SDOGeometry.java class. A switch case in every method allows us to compute the intersection, the area and the distance for specific shapes.

Note- The intersection function prints out coordinates in 3D space, and hence the Z coordinates for all the intersecting shapes will be 0.