Unit 13:
Geospatial Data, Indexing and Search

MarkLogic

Learning Objectives

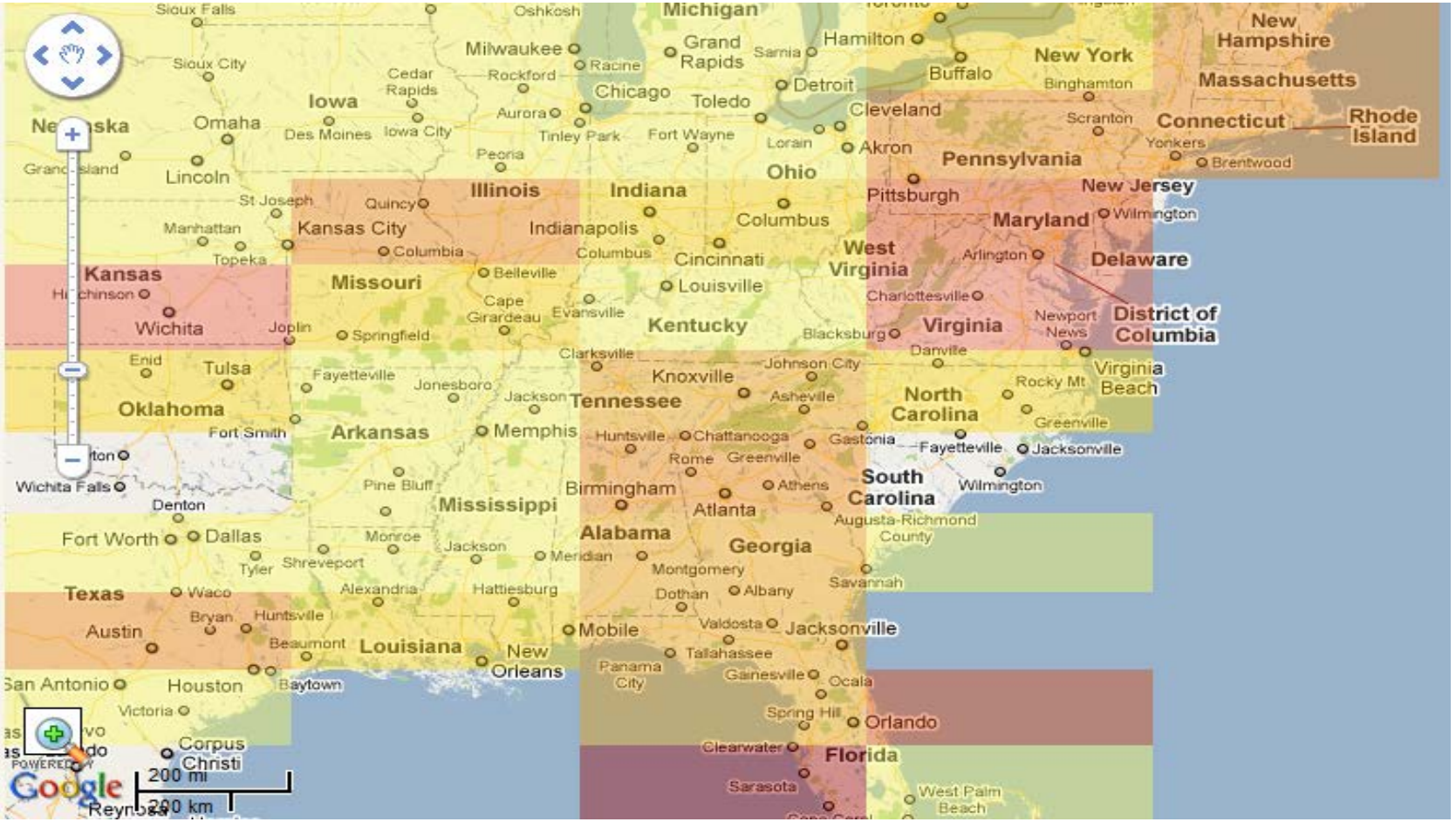
- Load geospatial data.
- Configure geospatial indexes.
- Build geospatial search queries.

MarkLogic

Examples

MarkLogic

Examples

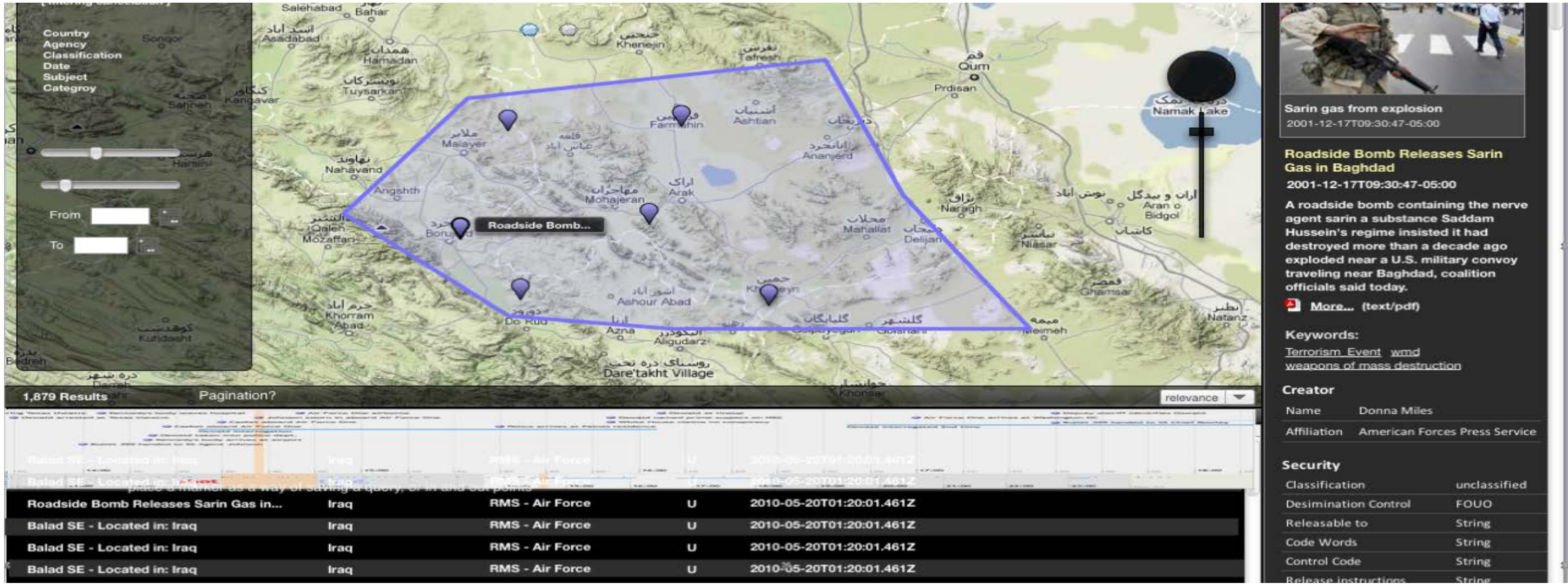


MarkLogic

Examples

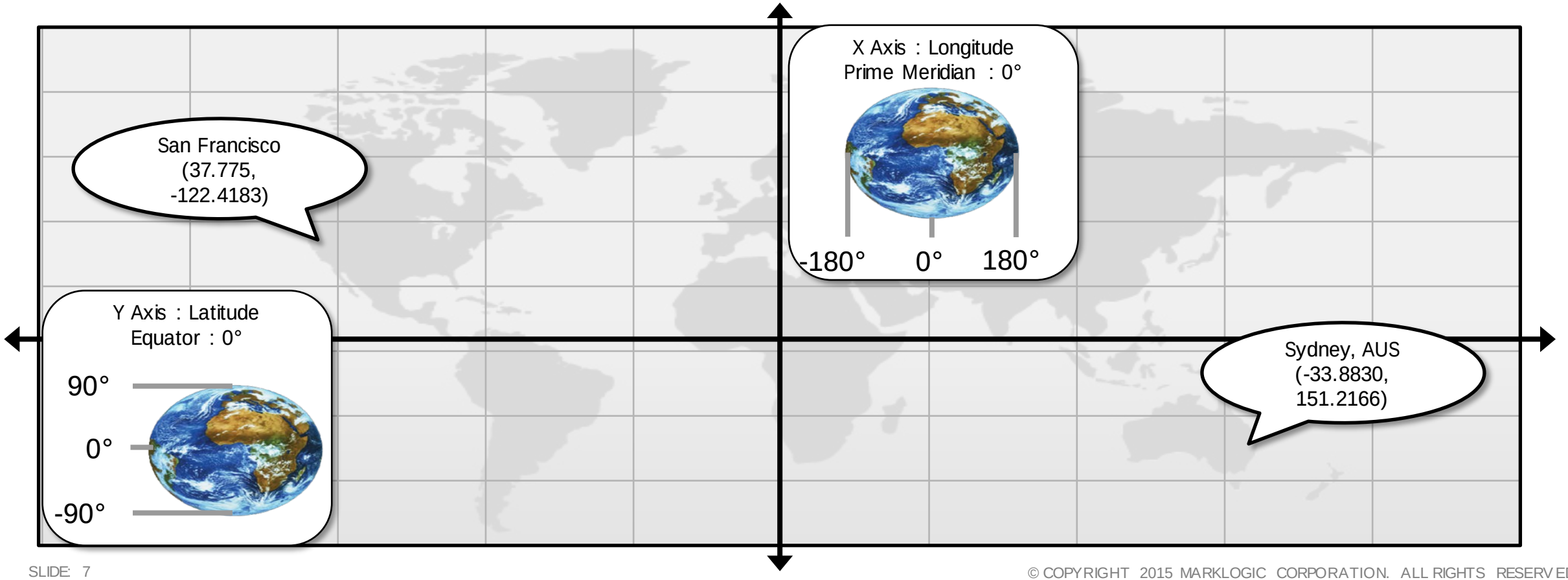
MarkLogic

Examples



Geospatial Data

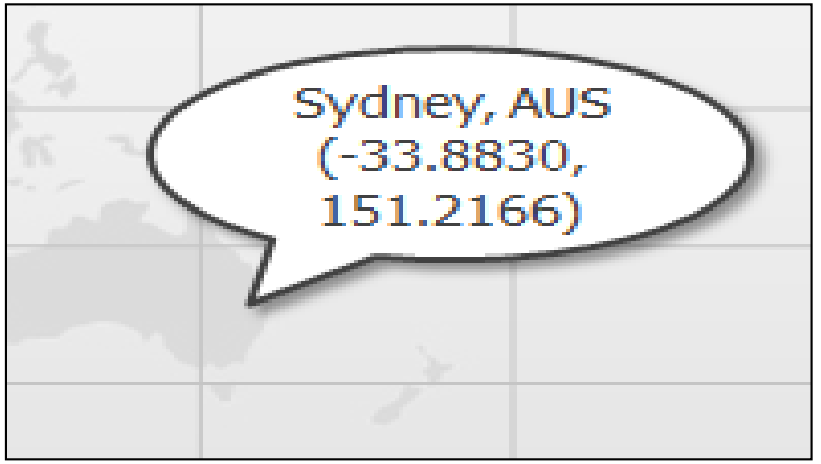
- A flat representation of a curved earth consisting of at least one ordered pair charted on the X and Y axis



MarkLogic

Geospatial Queries

- `cts.point()`
- An ordered pair containing a latitude and longitude
- The foundation for other geospatial shapes



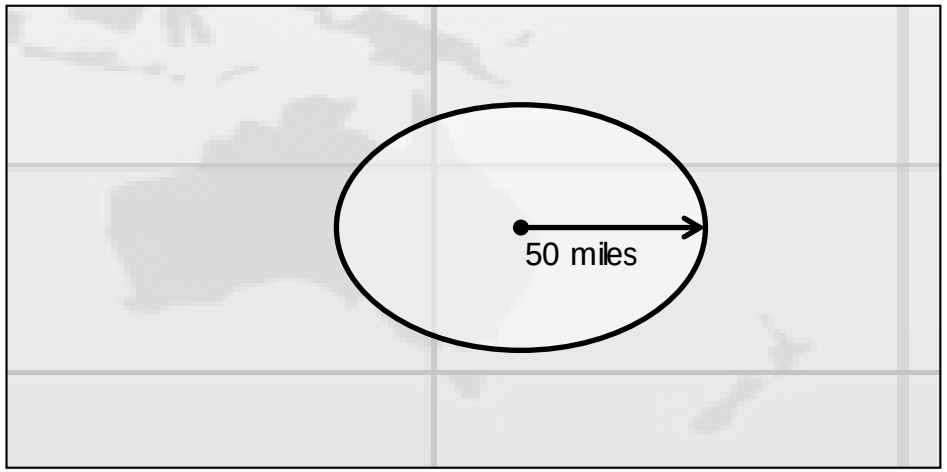

```
(: Define a point representing Sydney, Australia :)

cts.point(-33.8830, 151.2166)
```

MarkLogic

Geospatial Queries

- `cts.circle()`
- Defined by:
  - Specifying a center point
  - Specifying a radius measured in miles



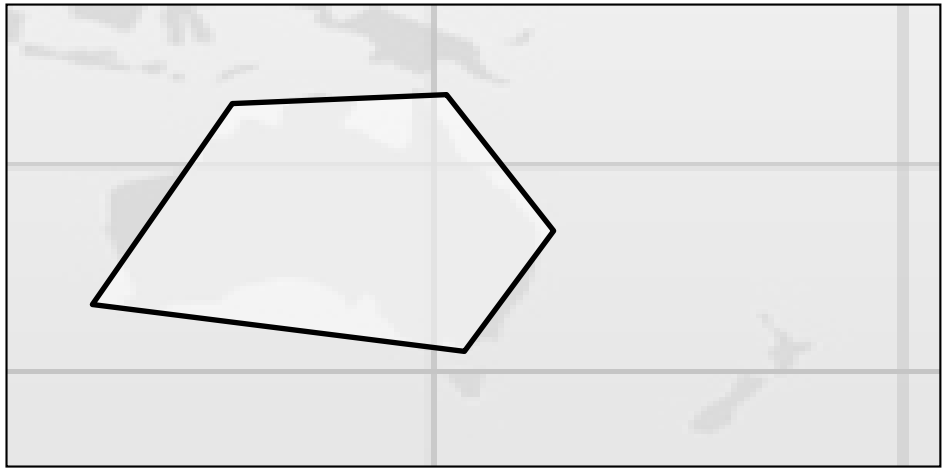

```
(: Define a circle radiating 50 miles from Sydney :)

cts.circle(50, cts:point(-33.8830, 151.2166))
```

MarkLogic

Geospatial Queries

- `cts.polygon()`
- Define a region with 3:n sided boundaries
- Composed using 3:n `cts.point()` definitions

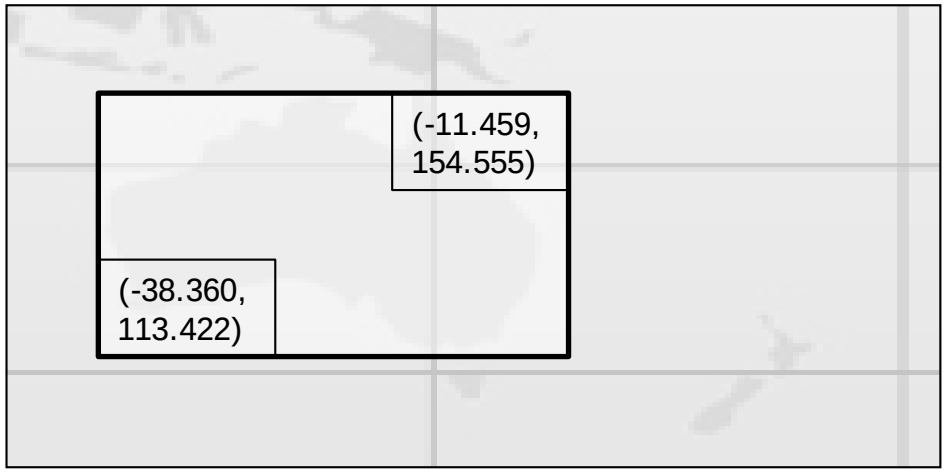


```
(: Define a general polygon over Australia :)
var points = [cts.point(-10.942, 144.799),
              cts.point(-25.900, 154.819),
              cts.point(-39.885, 146.755),
              cts.point(-35.111, 114.763),
              cts.point(-15.560, 120.454)]

cts.polygon(points)
```

Geospatial Queries

- `cts.box()`
- Defined by (in order):
  - South (Latitude)
  - West (Longitude)
  - North (Latitude)
  - East (Longitude)

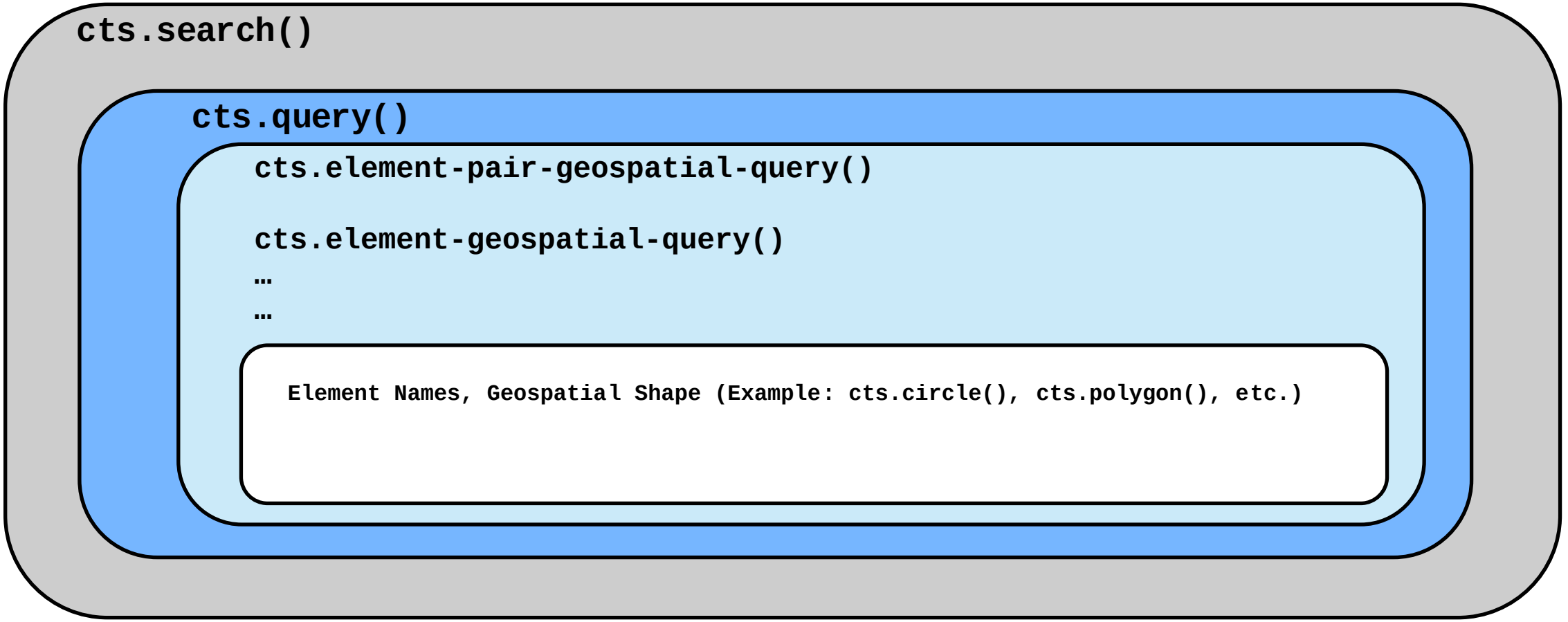



```
(: Define a general box over Australia :)

cts.box(-38.360, 113.422, -11.459, 154.555)
```

MarkLogic

Geospatial Search Queries



MarkLogic

Geospatial Search Queries (SJS)

```
(: SEARCH FOR DOCUMENTS IN A 100M RADIUS OF MIAMI :)

cts.search
  (
    cts.elementPairGeospatialQuery
      (
        xs.QName("Place"),
        xs.QName("Lat"),
        xs.QName("Lon"),
        cts.circle(100, cts.point(25.788969,-80.226439))
      )
  )
```

MarkLogic

Geospatial Search Queries (Node.js)

- Circle

```
dbRead.documents.query(
  qb.where(
    qb.geospatial(
      qb.geoProperty(qb.property("location"), qb.property("coordinates")),
      qb.circle(radius, lat, lon)
    )
  )
).result();
```

MarkLogic

Geospatial Search Queries (Node.js)

- Polygon

```
db.documents.query(
  qb.where(
    qb.geospatial(
      qb.geoProperty(qb.property("location"), qb.property("coordinates")),
      qb.polygon(shape)
    )
  )
).result();
```

MarkLogic

Geospatial Search Queries (Node.js)

- Polygon and text

```
db.documents.query(
  qb.where(
    qb.geospatial(
      qb.geoProperty(qb.property("location"), qb.property("coordinates")),
      qb.polygon(shape)
    )
  )
).result();
```

MarkLogic

Geospatial Indexes

- Purpose built indexes for maximum performance in working with geospatial data.
- Admin Tool→Configure→Databases→*Your DB*→Geospatial Indexes
  - Each index designed for different representations of the geospatial data.

MarkLogic

Geospatial Indexes

Geospatial Specific Indexes

Element

```
<element-name>
-38.36 113.42
</element-name>
```

```
{
“property-name”:
    [-38.36, 113.42]
}
```

Element Child

```
<parent-element-name>
  <child>
  -38.36 113.42
  </child>
</parent-element-name>
```

```
{
“parent-property-name”:
  {
    “property-name”:
      [-38.36, 113.42]
  }
}
```

Element Pair

```
<parent-element-name>
  <lat>
     -38.36
  </lat>
  <lon>
     113.42
  </lon>
</parent-element-name>
```

```
{
“parent-property-name”:
  {
    “lat”: “-38.36”,
    “lon”: “113.42”
  }
}
```

Attribute Pair

```
<parent-element-name
lat=“-38.36”
lon=“113.42”/>
```

```
Not applicable to JSON.
```

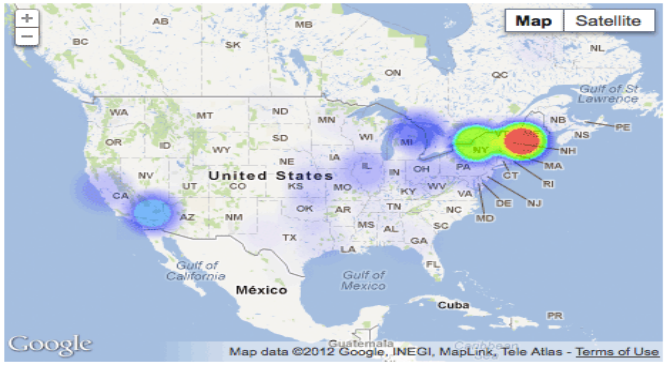
Map Widgets

- Use APIs for map integration, for example Google Maps, Earth, Yahoo Maps etc.
- Use HTML visualization widgets packaged with MarkLogic.

Points

- Pin individual locations
- Intelligent aggregation
- Interactively filter by drawing

Heatmaps



- Automatically calculate concentrations
- Grids or densities

GEOPHOTO

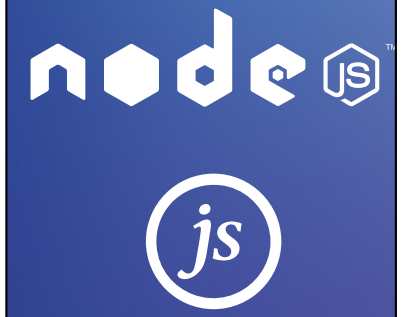

```
{
  "filename": "05.JPG",
  "location": {
    "type": "Point",
    "coordinates": [
      37.809333,
      -122.475833
    ]
  },
  "make": "Apple",
  "model": "iPhone 4",
  "created": 1315246591000,
  "binary": "/binary/05.JPG"
}
```


{
"filename": "05.JPG",
"location": {
"coordinates": [
37.809333,
-122.475833
]
},
"make": "Apple",
"model": "iPhone 4",
"created": 1315246591000,
"binary": "/binary/05.JPG"
}
This is good...


{
"type": "Point",
37.809333,
-122.475833
],
"city": "San Francisco",
"country": "United State
"created": 1315246591000,
"binary": "/binary/05.JPG"
}
This is better!!!

```
{
  "filename": "05.JPG",
  "location": {
    "type": "Point",
    "coordinates": [
      37.809333,
      -122.475833
    ],
    "city": "San Francisco",
    "country": "United States"
  },
  "make": "Apple",
  "model": "iPhone 4",
  "created": 1315246591000,
  "binary": "/binary/05.JPG"
}
```


What do we know about...




```
<?xml version="1.0" encoding="UTF-8"?>
<sem:triples xmlns:sem="http://marklogic.com/semantics">
 <sem:triple>
  <sem:subject>http://dbpedia.org/resource/United_States</sem:subject>
  <sem:predicate>http://xmlns.com/foaf/0.1/homepage</sem:predicate>
  <sem:object>http://www.usa.gov/</sem:object>
 </sem:triple>
 <sem:triple>
  <sem:subject>http://dbpedia.org/resource/United_States</sem:subject>
  <sem:predicate>http://dbpedia.org/ontology/anthem</sem:predicate>
  <sem:object>http://dbpedia.org/resource/The_Star-Spangled_Banner</sem:object>
 </sem:triple>
 <sem:triple>
  <sem:subject>http://dbpedia.org/resource/United_States</sem:subject>
  <sem:predicate>http://dbpedia.org/ontology/capital</sem:predicate>
  <sem:object>http://dbpedia.org/resource/Washington,_D.C.</sem:object>
 </sem:triple>
```

https://github.com/marklogic/Geophoto

Labs: Unit 13

Exercise 1: Configure Geospatial Indexes
Exercise 2: Load Geospatial Data
Exercise 3: Build Geospatial Search Queries

Unit Review Question 1:

In MarkLogic, geospatial search can combine with full text search, range, semantic, and reverse queries:

1. True
2. False

Unit Review Question 1:

In MarkLogic, geospatial search can combine with full text search, range, semantic, and reverse queries:

1. **True**
2. False

Unit Review Question 2:

When performing a geospatial search using the circle shape, the distance of the radius is specified in:

1. Miles
2. Kilometers
3. Light years
4. Furlongs

Unit Review Question 2:

When performing a geospatial search using the circle shape, the distance of the radius is specified in:

1. **Miles**
2. Kilometers
3. Light years
4. Furlongs

Unit Review Question 3:

When defining a geospatial index, the concept of namespaces and attributes apply to:

1. JSON
2. XML
3. Binary
4. Full text

Unit Review Question 3:

When defining a geospatial index, the concept of namespaces and attributes apply to:

1. JSON
2. **XML**
3. Binary
4. Full text