Quality-Driven Geospatial Data Integration

Snehal Thakkar & Craig Knoblock
University of Southern California

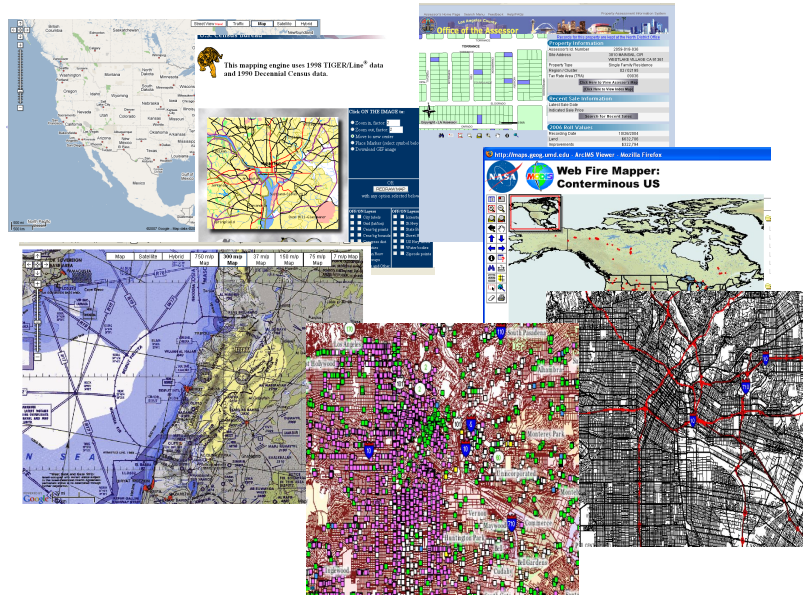
November 8, 2007

Outline

- Introduction & Motivation
- Assessing the Quality of Sources
- Representing Content and Quality
- Exploiting Quality in Query Answering
- Experimental Evaluation
- Related work
- Conclusions

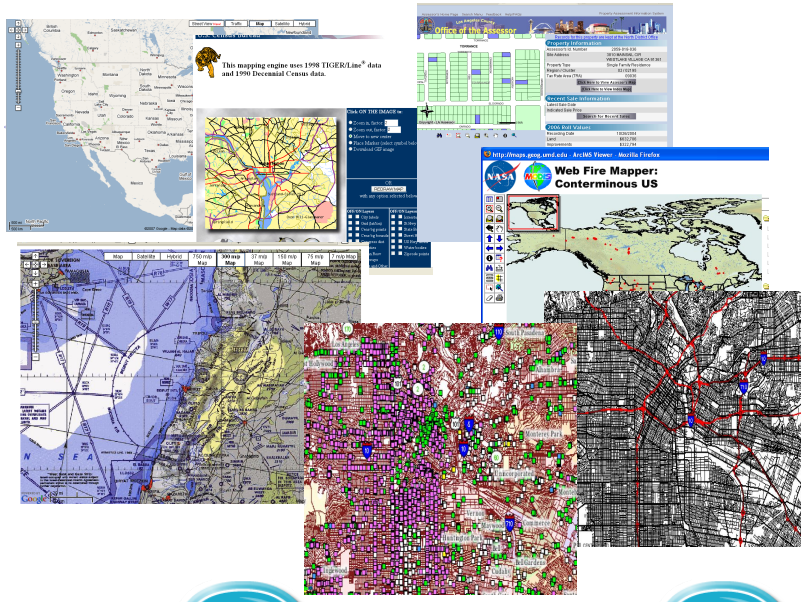
Introduction & Motivation

- Many disaster response and urban planning require integrated view of geospatial data



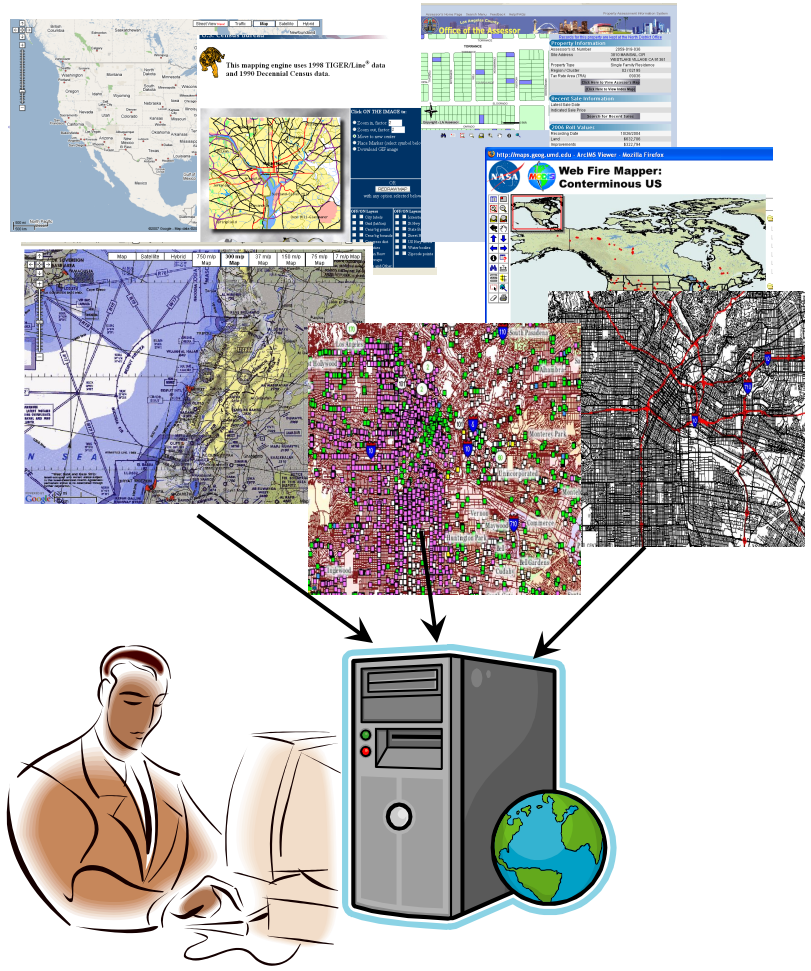
Introduction & Motivation

- Many disaster response and urban planning require integrated view of geospatial data
- Manually integrating geospatial data from a large number of sources is very hard



Introduction & Motivation

- Many disaster response and urban planning require integrated view of geospatial data
- Manually integrating geospatial data from a large number of sources is very hard
- There is a need for a geospatial data integration framework that
  - Automatically generates representations of sources
  - Dynamically provides high quality data



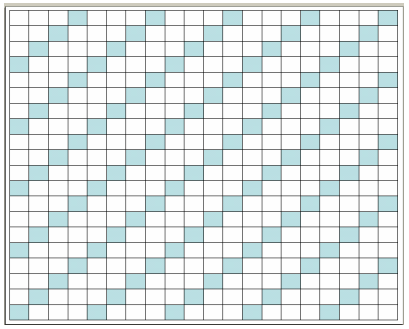
Motivating Examples

- Find road vector data covering the bounding box `[[33,-115],[34,-116]]' with completeness over 50%
- Find road vector data covering the bounding box `[[33,-115],[34,-116]]' with the highest completeness
- Find satellite image and road vector data covering bounding box `[[33,-116], [34,-117]]' such that both the resolution and date differences are minimized

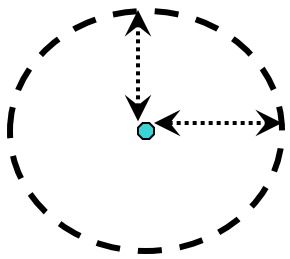
Outline

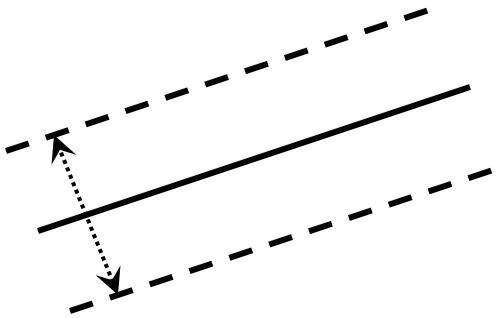
- Introduction & Motivation
- Assessing the Quality of Sources
- Representing Content and Quality
- Exploiting Quality in Query Answering
- Experimental Evaluation
- Related work
- Conclusions

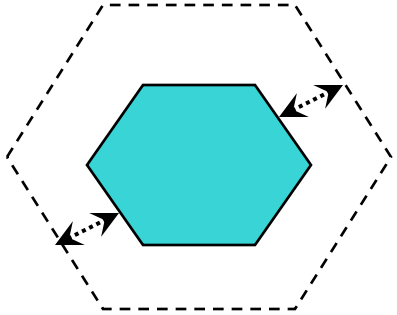
Estimating Vector Quality



- Sample data from known and new source
- Compute value for completeness and positional accuracy attributes
  - Completeness
    - $\#features_{(new)} * completeness_{(known)} / \# features_{(known)}$
  - Accuracy bounds
    - Use accuracy bounds of the known sources
  - Features within accuracy bounds
    - # of features that fall within accuracy bounds/ # features





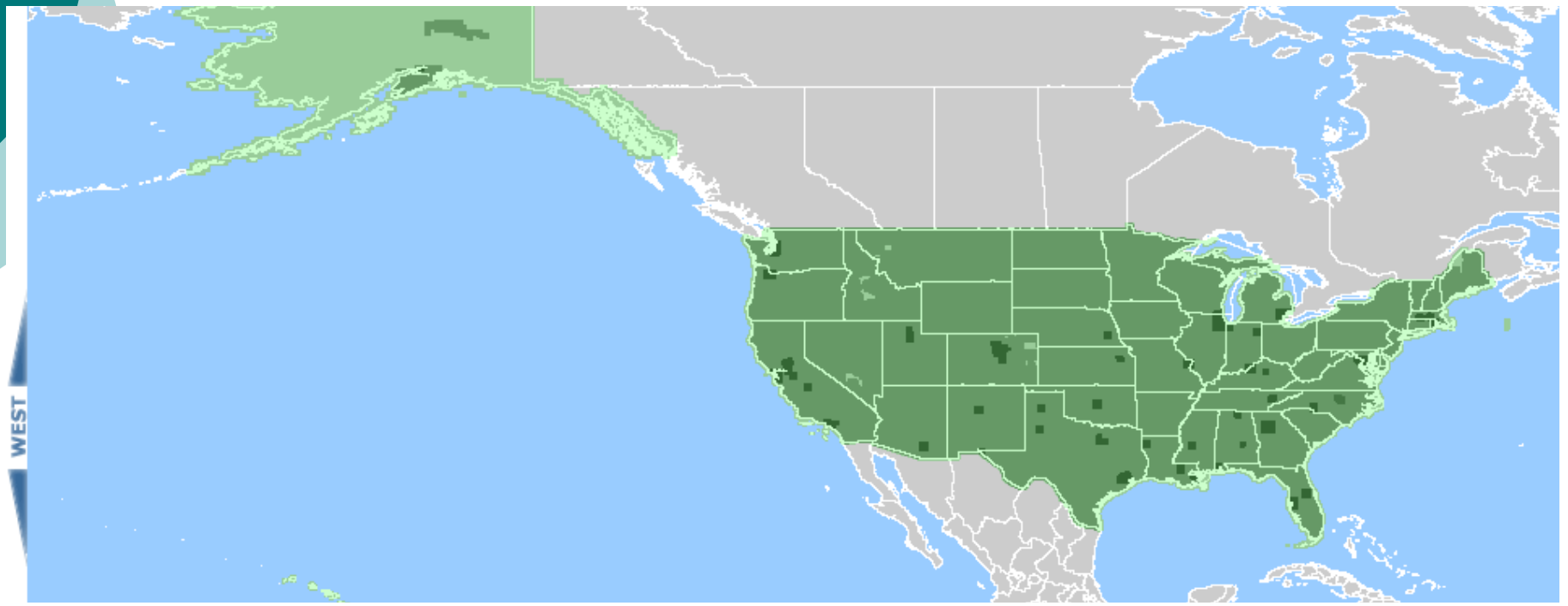


Experimental Quality Results

- Used 1268 shapefiles containing different vector sources

Type	% Data	Avg. Comp. & Acc. Without Sampling		Avg. % Error with Sampling	
		Completeness	Accuracy	Completeness	Accuracy
Point	10	91.76	95.6	17.54	12.27
Point	20	91.76	95.6	14.27	7.95
Polyline	10	38.09	80.28	24.69	8.68
Polyline	20	38.09	80.28	20.74	7.95
Polygon	10	68.12	87.15	25.01	11.20
Polygon	20	68.12	87.15	20.51	10.97

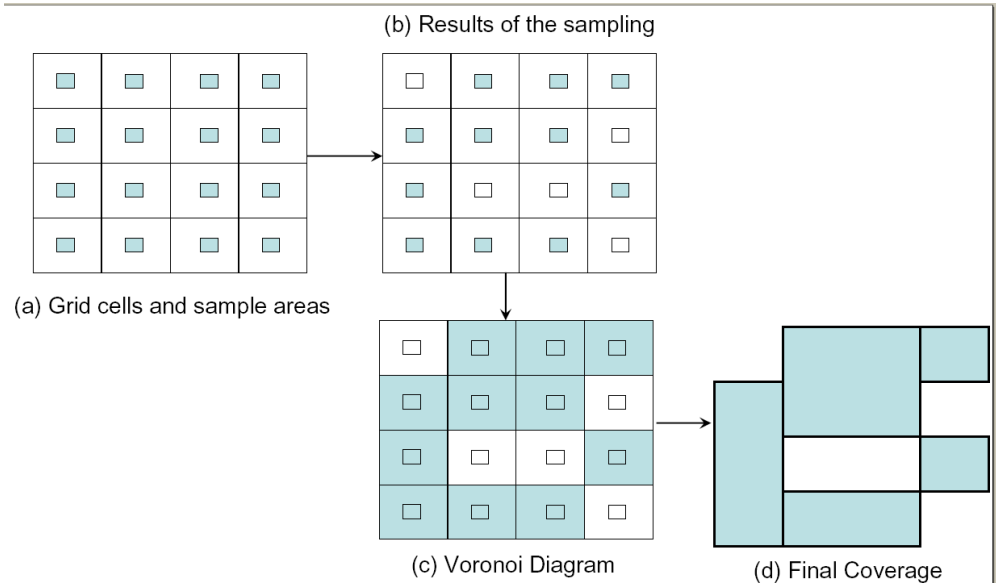
Raster Quality Estimation: Overstated Coverage



- Water no coverage
- Land no coverage
- B/W Satellite Image Only
- Topo maps & B/W Satellite Image
- Multi-spectral Satellite Image

Estimating Raster Coverage & Completeness

- Address the problem of sources overstating coverage
- Sample data from a source
- Use the sampling results and Voronoi diagram
- Estimate accurate coverage and completeness



Experimental Results: Raster Coverage Estimation

- Automatic estimation of Raster Quality
  - 60 queries with resolutions 1,5,10,50 m/p
  - Compare reported coverage with estimated coverage by sampling
  - Estimated coverage
    - loses some images (lower recall)
    - returns fewer empty images (higher precision)

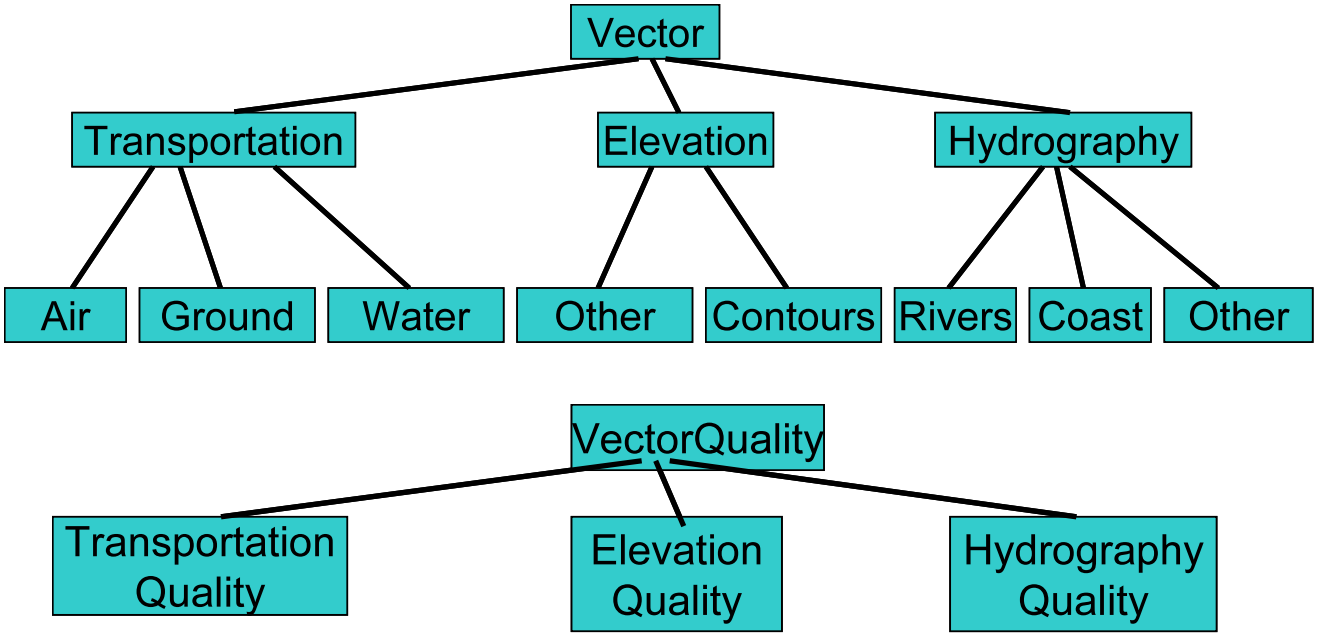
Resolution	Reported Coverage			Estimated Coverage		
1 meter/pixel	Precision	Recall	F-measure	Precision	Recall	F-measure
1	72.15	100.00	83.82	94.12	84.21	88.89
5	82.43	100.00	90.37	91.38	86.89	89.08
10	81.82	100.00	90.00	92.86	86.67	89.66
15	81.58	100.00	89.86	90.00	87.10	88.52
Total	79.23	100.00	88.41	91.94	86.22	88.99

Outline

- Introduction & Motivation
- Assessing the Quality of Sources
- Representing Content and Quality
- Exploiting Quality in Query Answering
- Experimental Evaluation
- Related work
- Conclusions

Representation: Domain Concepts

- Content
  - Set of domain concepts by merging FGDC, NGA, and NationalMap concepts
- Quality
  - Similar hierarchy for quality of data for each domain concept
    - E.g. Road → RoadQuality



Representation: Domain Concept Attributes

- Vector
  - source, type, format, cs, bbox, vectorobj
- Raster
  - source, type, format, cs, bbox, size, resolution, rasterobj
- VectorQuality
  - source, type, date, completeness, resolution, horizontalaccuracy, verticalaccuracy, vectorswithinaccuracybounds, attributecompleteness
- RasterQuality
  - source, type, date, completeness, originalresolution, multispectral

Representation: Source Descriptions

- Source represented using two relations
  - Content
  - Quality
- Datalog descriptions
  - Content
    - Type of data: domain relation in the body
    - Coverage specified using constraints with spatial operations

```
NavteqRoads(bbox, vectorobj):-
    Roads(type, format, cs,
          bbox, source, vectorobj)^
    bbox coveredby
          `[[33,-117],[34,-118]]'^
    source = `Navteq'^
    type = `Roads'^
    format = `Shapefile'^
    cs = `EPSG:4326'
```

Representation: Source Descriptions

- Source represented using two relations
  - Content
  - Quality
- Datalog descriptions
  - Content
    - Type of data: domain relation in the body
    - Coverage specified using constraints with spatial operations
  - Quality
    - Facts specifying the quality
    - Rule defining the relationship with corresponding quality relation

```
NavteqRoadsQuality(res, date,
    horiz-acc, vert-acc,
    vectorsin-acc-bounds,
    attr-comp, completeness):-
  RoadQuality(source, type, res, date,
    horiz-acc, vert-acc, vectorsin-acc-
    bounds, attr-comp, completeness)^
  source = `Navteq'^
  type = `Roads'

NavteqRoadsQuality(5,1/1/2005,3.6,
        3.6, 85%, 90%, 96%)
```

Representation: Queries

- Expressed by Datalog rules
- Three parts: data, quality, combination
  - Predicates allowed
    - Domain relations
    - Operations
      - Spatial selection
        - intersects, coveredby, disjoint
      - Aggregate
        - pack, unpack, sum, average,
        - min, max, skylinemin, skylinemax
      - Mathematical
        - add, subtract, multiply, divide
      - Order Constraints
        - e.g. completeness > 50

Representation: Sample Query 1

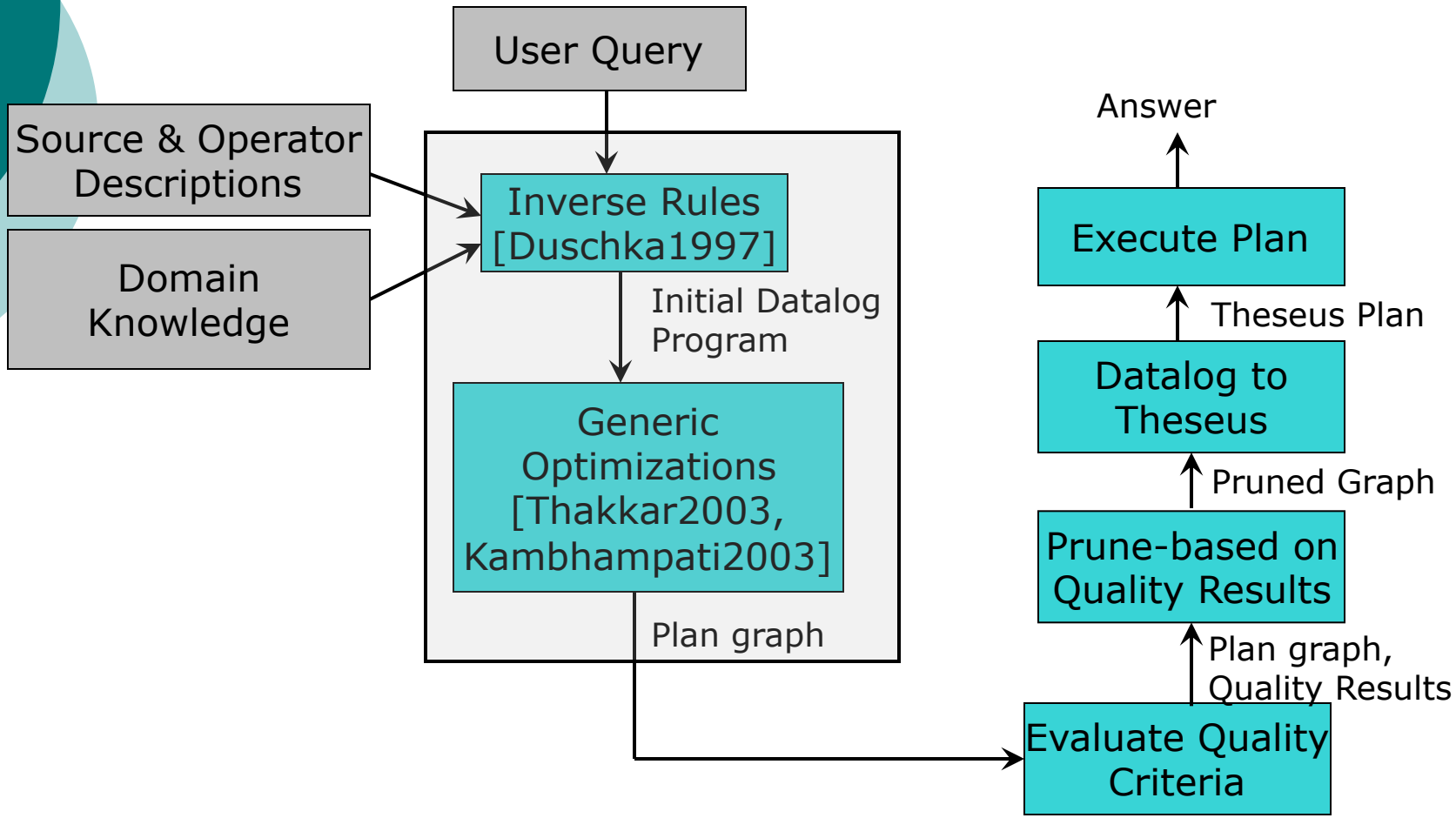
- Find road vector data covering the bounding box `[[33,-115],[34,-116]]' with completeness over 50%

```
Q1(vectorobj, completeness):-
   Q1Data(type, source, vectorobj)^
   Q1Quality(type, source, completeness)

Q1Data(type, source, vectorobj):-
   Roads(type, format, cs, bbox, source, vectorobj)^
   bbox coveredby `[[33,-115],[34,-116]]'

Q1Quality(type, source, completeness):-
   RoadQuality(source, type, res, date, horiz-acc,
         vert-acc, vectorsin-acc-bounds, attr-comp,
         completeness)^
   completeness > 50
```

Outline

- Introduction & Motivation
- Assessing the Quality of Sources
- Representing Content and Quality
- Exploiting Quality in Query Answering
- Experimental Evaluation
- Related work
- Conclusions

QGM's Query Answering



Inverse Rules [Duschka 1997]

- Determine how to query domain relations
- Invert the source descriptions
- In the example query
  - Definition of Roads & SatelliteImage as views over sources
  - Definition of RoadQuality and SatelliteImageQuality as views over source quality

```
NavteqRoads(bbox, vectorobj):-
    Roads(type, format, cs,
        bbox, source, vectorobj)^
    bbox coveredby
        `[[33,-116],[34,-117]]'^
    source = `Navteq'^
    type = `Roads'^
    format = `Shapefile'^
    cs = `EPSG:4326'
```

```
Roads(`Roads', `Shapefile',
    `EPSG:4326', bbox, `Navteq',
    vectorobj):-
  NavteqRoads(bbox, vectorobj)^
  bbox coveredby
        `[[33,-116],[34,-117]]'
```

Datalog Program Generation

- Identify Relevant Rules
  - Extension: Check geospatial constraints
  - Find sources that
    - Appear in definition of relevant domain concepts
    - Do not have conflicting coverage constraints

```
Roads(`Roads', `Shapefile',
    `EPSG:4326', bbox, `Navteq',
    vectorobj):-
  NavteqRoads(bbox, vectorobj)^
  bbox coveredby
        `[[33,-116],[34,-117]]'

Roads(`Roads', `Shapefile',
    `EPSG:4326', bbox, `Navteq',
    vectorobj):-
  TigerRoads(bbox, vectorobj)^
  bbox coveredby
    `[[33,-116],[34,-117]]'

Parks(`Roads', `Shapefile', `EPSG:
    4326', bbox, `Navteq',
    vectorobj):-
  NGAParks(bbox, vectorobj)^
  bbox coveredby
    `[[33,-116],[34,-117]]'
```

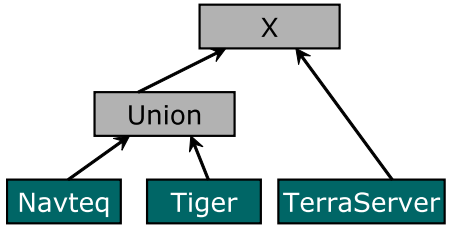
Datalog Program Generation

- Identify Relevant Rules
  - Extension: Check geospatial constraints
  - Find sources that
    - Appear in definition of relevant domain concepts
    - Do not have conflicting coverage constraints

```
Roads(`Roads', `Shapefile',
    `EPSG:4326', bbox, `Navteq',
    vectorobj):-
  NavteqRoads(bbox, vectorobj)^
  bbox coveredby
        `[[33,-116],[34,-117]]'

Roads(`Roads', `Shapefile',
    `EPSG:4326', bbox, `Navteq',
    vectorobj):-
  TigerRoads(bbox, vectorobj)^
  bbox coveredby
    `[[33,-116],[34,-117]]'

Parks(`Roads', `Shapefile', `EPSG:
    4326', bbox, `Navteq',
    vectorobj):-
  NGAParks(bbox, vectorobj)^
  bbox coveredby
    `[[33,-116],[34,-117]]'
```

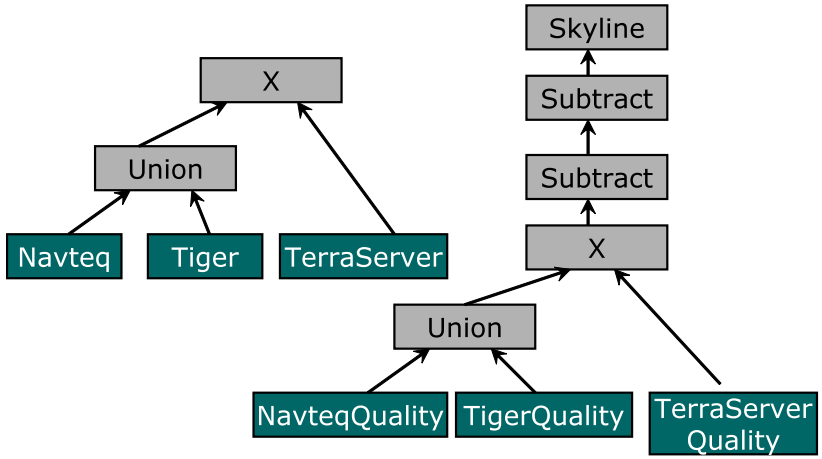
Generated Plan

- Two branches
  - Content
    - Has requests to data sources
    - Select operations to apply constraints
  - Quality
    - Has requests to obtain facts about quality of data for sources that appear in the content plan
    - May have requests to mathematical, aggregate, or skyline operations
  - In our example query
    - Assume two relevant vector sources
    - Assume one image source



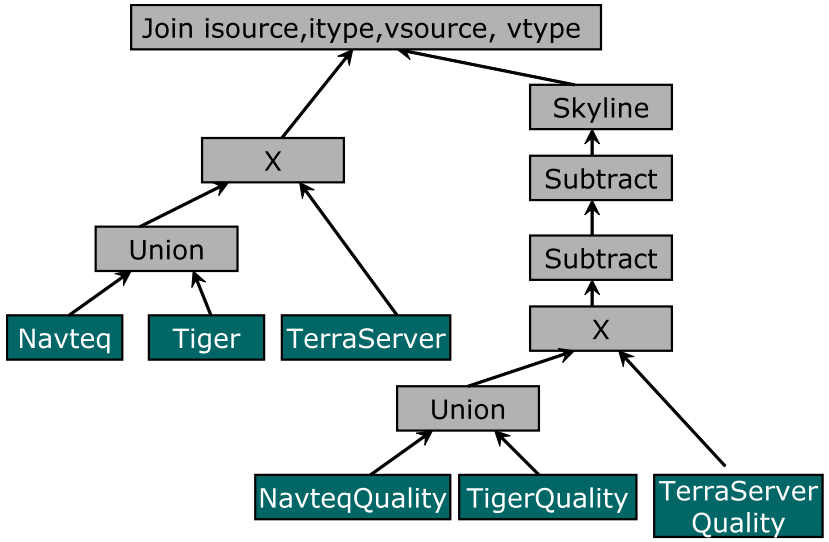
Generated Plan

- Two branches
  - Content
    - Has requests to data sources
    - Select operations to apply constraints
  - Quality
    - Has requests to obtain facts about quality of data for sources that appear in the content plan
    - May have requests to mathematical, aggregate, or skyline operations
  - In our example query
    - Assume two relevant vector sources
    - Assume one image source

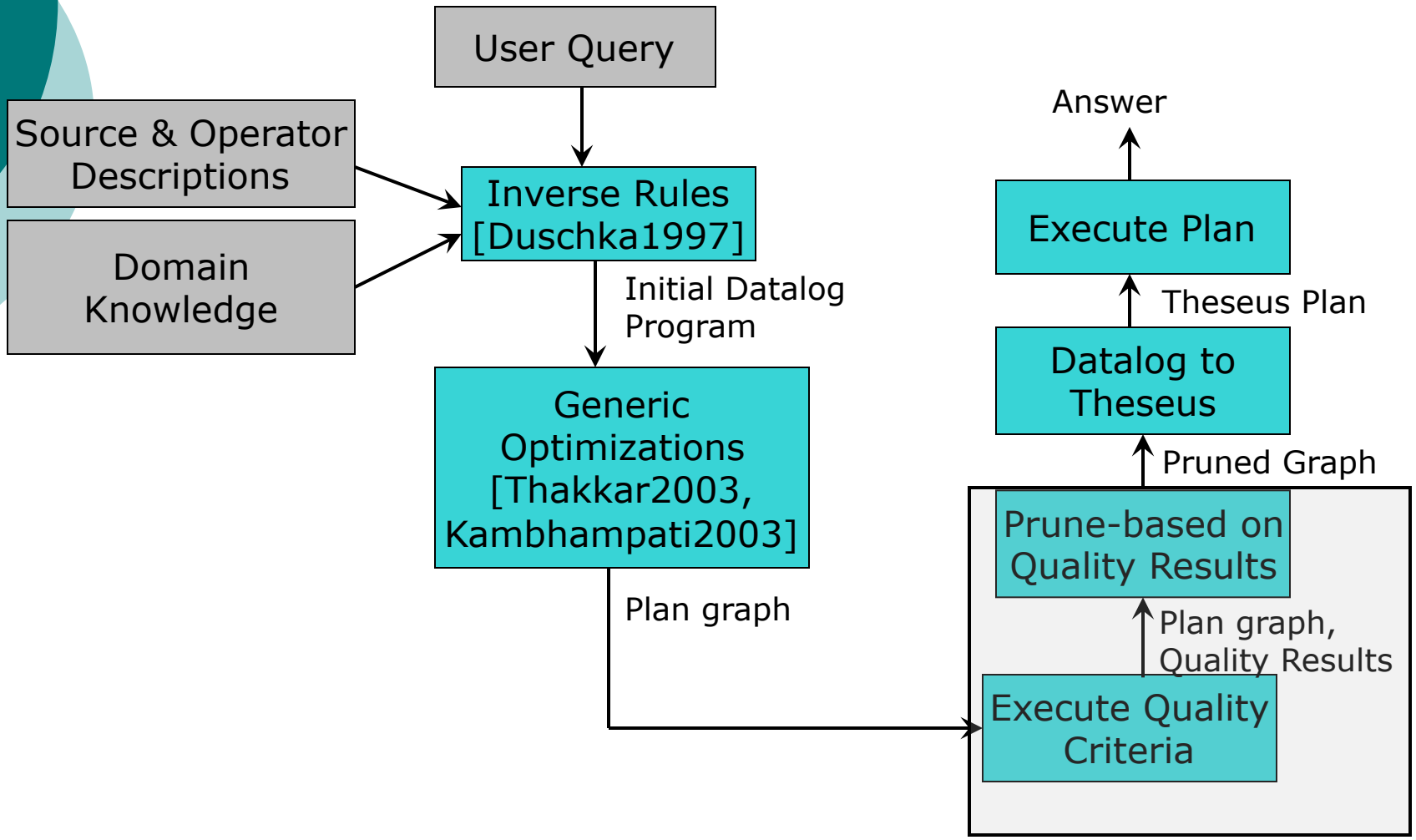


Generated Plan

- Two branches
  - Content
    - Has requests to data sources
    - Select operations to apply constraints
  - Quality
    - Has requests to obtain facts about quality of data for sources that appear in the content plan
    - May have requests to mathematical, aggregate, or skyline operations
  - In our example query
    - Assume two relevant vector sources
    - Assume one image source

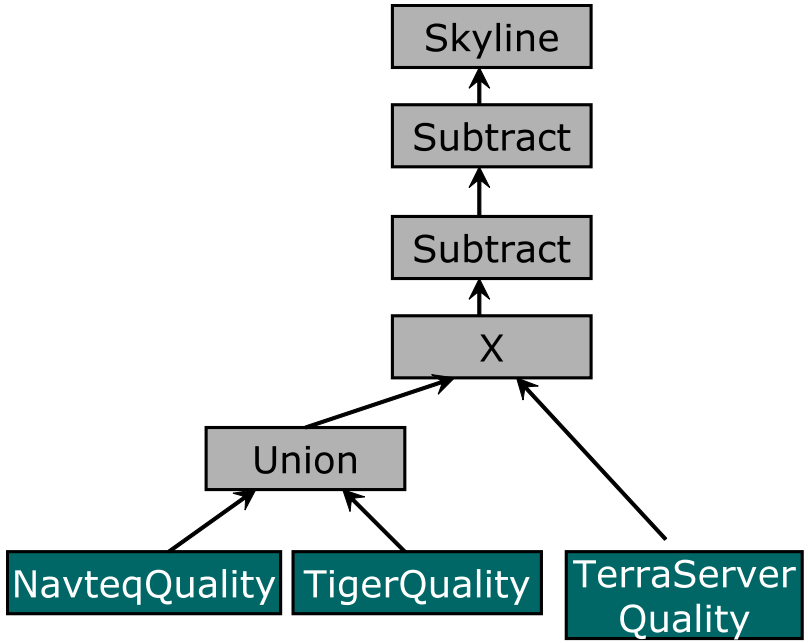


QGM's Query Answering



Executing Quality Criteria

- Obtain Quality facts
- Apply necessary relational, mathematical, or aggregate operations
- Apply constraints and/or skyline operations
- Resulting tuples include source name and type for each type of data and any other attributes requested in quality query

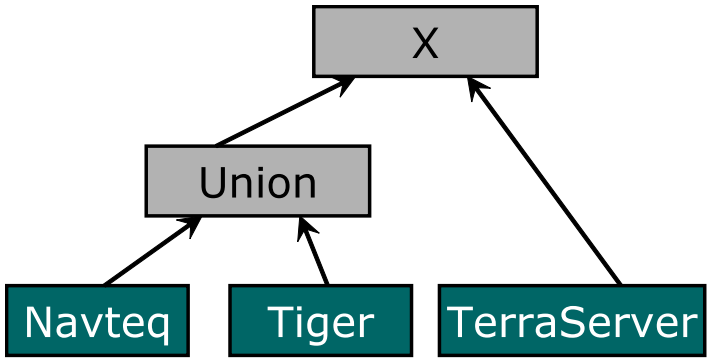


Prune based on Quality Results

- Remove all sources that did not satisfy quality criteria
  - If a source(S1) has completeness 20% and quality criteria is completeness > 50%
  - Remove source (S1) from the content subtree
- Check join constraints in the graph connected to quality subtree
  - Remove branches that do not produce tuples

Combination	Resdiff (m/p)	Datediff (days)
Navteq & TS	5	365
Tiger & TS	12	1825

Quality Statstics

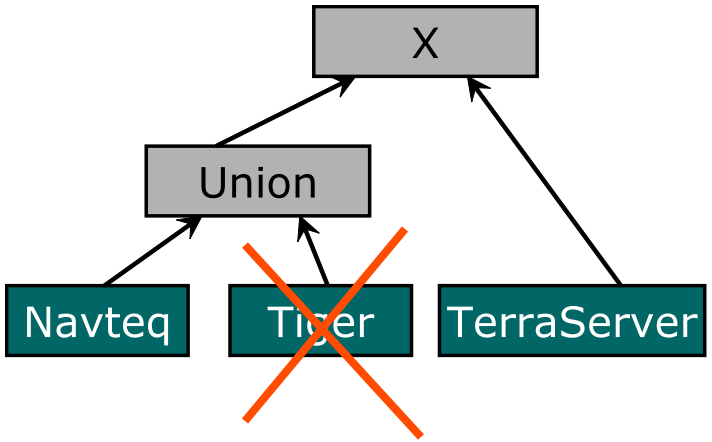


Prune based on Quality Results

- Remove all sources that did not satisfy quality criteria
  - If a source(S1) has completeness 20% and quality criteria is completeness > 50%
  - Remove source (S1) from the content subtree
- Check join constraints in the graph connected to quality subtree
  - Remove branches that do not produce tuples

Combination	Resdiff (m/p)	Datediff (days)
Navteq & TS	5	365
Tiger & TS	12	1825

Quality Statstics

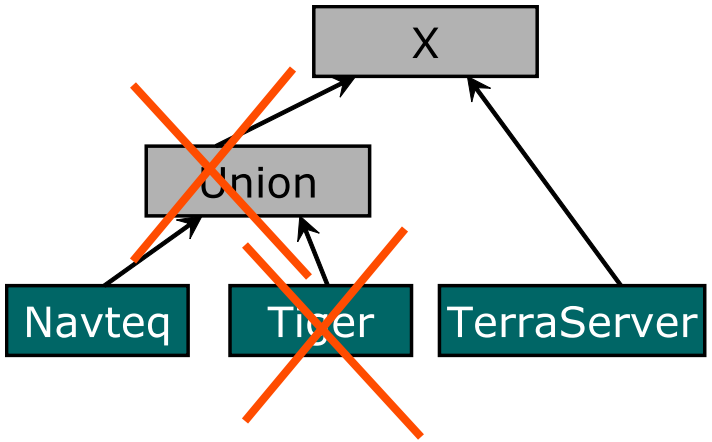


Prune based on Quality Results

- Remove all sources that did not satisfy quality criteria
  - If a source(S1) has completeness 20% and quality criteria is completeness > 50%
  - Remove source (S1) from the content subtree
- Check join constraints in the graph connected to quality subtree
  - Remove branches that do not produce tuples

Combination	Resdiff (m/p)	Datediff (days)
Navteq & TS	5	365
Tiger & TS	12	1825

Quality Statstics



Outline

- Introduction & Motivation
- Assessing the Quality of Sources
- Representing Content and Quality
- Exploiting Quality in Query Answering
- Experimental Evaluation
- Related work
- Conclusions

Experimental Evaluation

- Setup
  - Dual Xeon processor, 3 GB memory
    - Actual use: half processor, 1GB memory
  - Data sources
    - Real-world shapefiles, ArcIMS services, and Web Map Services
- Method
  - Compare with Prometheus
    - Data integration system that supports geospatial data without any quality information
  - Compare
    - Quality
    - Response time

Query Answering: Quality of Answers

- Query answering
  - Quality
    - One standard deviation better in completeness for most queries
    - Half standard deviation better for accuracy

Type	QGM		Average		Std. Deviation	
	% Comp.	% Acc.	% Comp.	% Acc.	% Comp.	% Acc.
Constraint	59.81	87.61	47.71	83.12	17.36	9.31
Aggregate	68.19	89.97	47.71	83.12	17.36	9.31
Skyline	64.03	87.90	47.71	83.12	17.36	9.31

Query Answering: Response Time

Query	# of Sources	Prometheus					QGM				
		Time in Seconds					Time in Seconds				
		Gen.	Opt.	Exec.	Total	# results	Gen.	Opt.	Exec.	Total	# results
Constraint	0-5	32	98	126	256	3.7	32	109	113	254	3.2
Constraint	5-10	33	119	279	431	7.9	33	116	196	345	5.8
Constraint	10-20	32	131	872	1035	16.1	32	138	524	694	11.2
Constraint	20-30	34	168	1985	2187	24.3	34	159	871	1064	17.6
Aggregate	0-5	32	98	126	256	3.7	32	113	102	247	1.3
Aggregate	5-10	33	119	279	431	7.9	33	116	115	264	2.1
Aggregate	10-20	32	131	872	1035	16.1	32	137	167	336	3.7
Aggregate	20-30	34	168	1985	2187	24.3	34	162	190	386	4.1
Skyline	0-5	32	98	126	256	3.7	32	140	134	306	2.9
Skyline	5-10	33	119	279	431	7.9	33	192	184	409	4.6
Skyline	10-20	32	131	872	1035	16.1	32	297	372	701	7.2
Skyline	20-30	34	168	1985	2187	24.3	34	421	579	1034	9.8

Outline

- Introduction & Motivation
- Assessing the Quality of Sources
- Representing Content and Quality
- Exploiting Quality in Query Answering
- Experimental Evaluation
- Related work
- Conclusions

Related Work

- Geospatial Data Integration
  - Hermes [Adali95], MIX [Gupta99], GeonGrid [Manipura03], VirGIS [Boucelma04]
    - Focus access methods and formats
    - GeonGrid also has some quality and ontology components
  - ODGIS [Fonesca 02], GSA [Arpinar 06], SWING [Klien06]
    - Creation ontology for geospatial data and matching data layers
- Quality-driven data integration
  - Biological data [Eckman06, Mihila05]
    - Focus is on completeness
  - General-purpose [Neumann01, Bleiholder2006, Scannapieco05]
    - Assign one quality score based on user-supplied weights
- QGM
  - A Geospatial mediation framework that supports quality
  - Automatic generation of descriptions
  - More expressive quality criteria specification

Outline

- Introduction & Motivation
- Assessing the Quality of Sources
- Representing Content and Quality
- Exploiting Quality in Query Answering
- Experimental Evaluation
- Related work
- Conclusions

Contributions

- Algorithms to automatically estimate the quality of data provided by geospatial data sources
- A declarative specification of both the content and the quality of geospatial sources
- A quality-driven query answering algorithm