RASED: A Scalable Dashboard for Monitoring Road Network Updates in OSM

Mashaal Musleh, Mohamed F. Mokbel

University of Minnesota

June 2022

UNIVERSITY OF MINNESOTA
Driven to Discover®

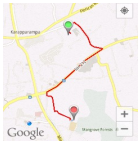
Outline

1. **Motivation**
2. **RASED Queries**
3. **RASED Architecture & Main Modules**
   - Data Collection & Processing
   - Storage & Indexing
   - Query Execution
   - User Interface
4. **Experimental Results**
5. **RASED in Action (A Quick Demo)**

Outline

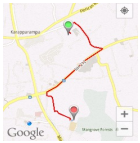
1. **Motivation**
2. **RASED Queries**
3. **RASED Architecture & Main Modules**
   - Data Collection & Processing
   - Storage & Indexing
   - Query Execution
   - User Interface
4. **Experimental Results**
5. **RASED in Action (A Quick Demo)**

Road Network Queries…



Range Query Routing …

Road Network Queries…

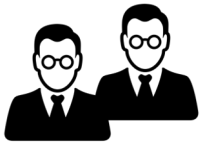


Range Query

Routing

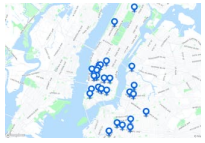
…

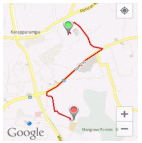
Map Analyzers



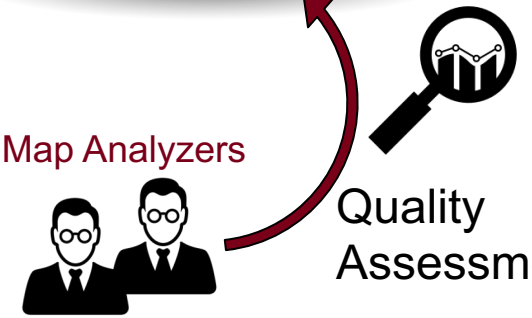
Quality Assessment

Road Network Queries…



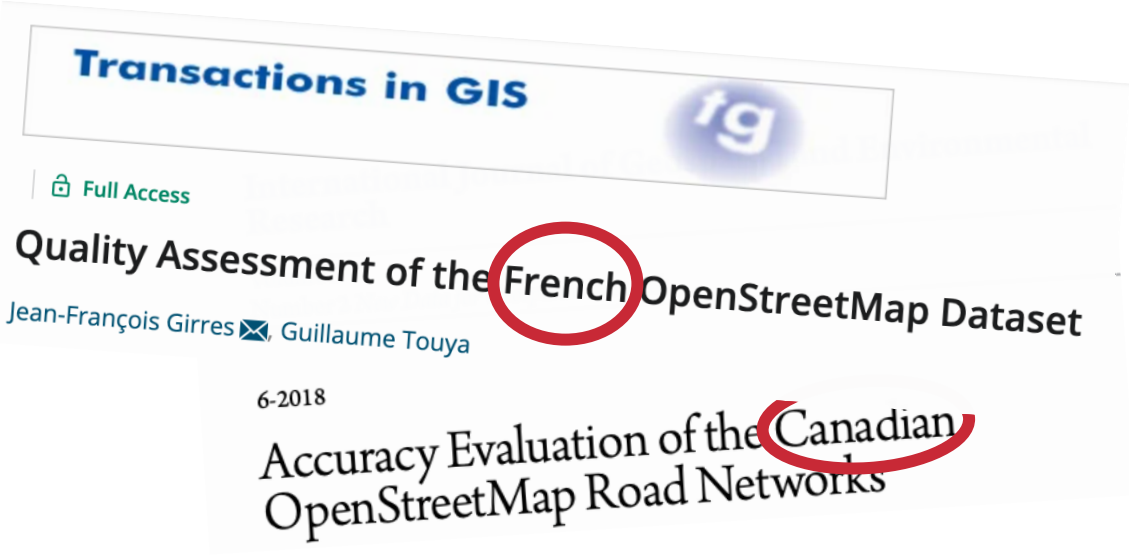


Range Query Routing …

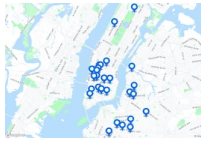


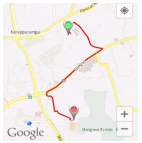
Map Analyzers

Quality Assessment



Road Network Queries…

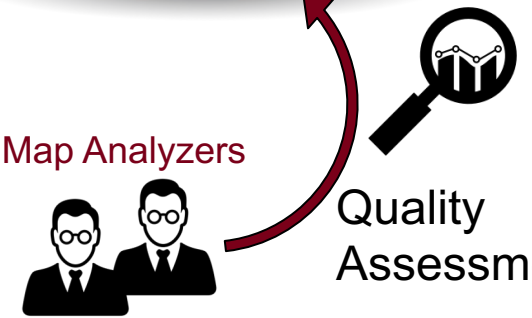




Range Query Routing …

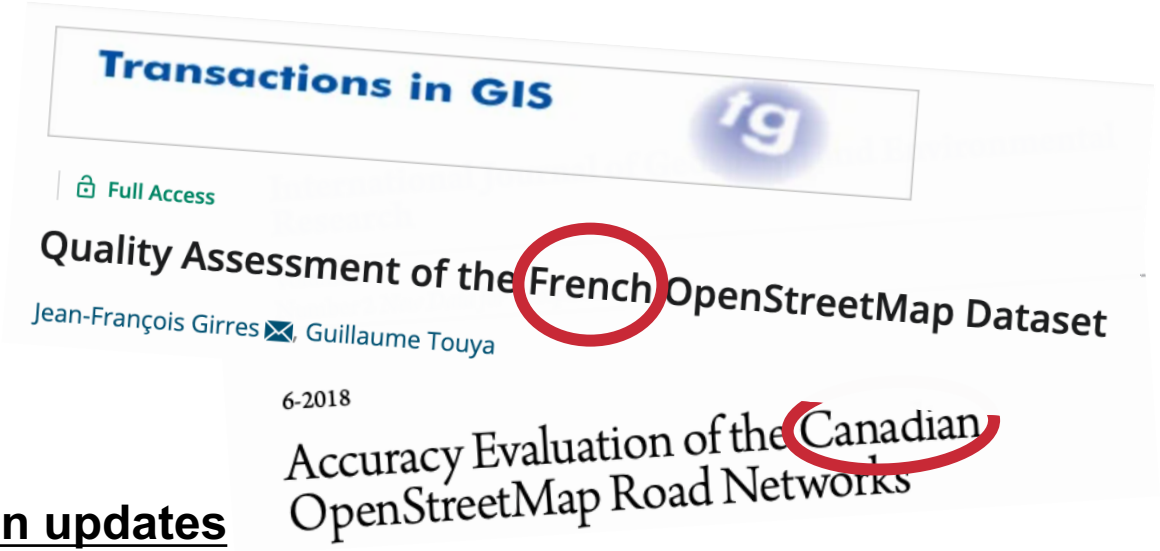
OSM:

- **12+ Billion updates**
- **3 TB Data**

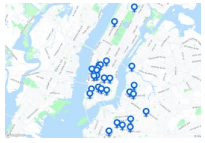


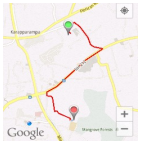
Map Analyzers

Quality Assessment



Road Network Queries…

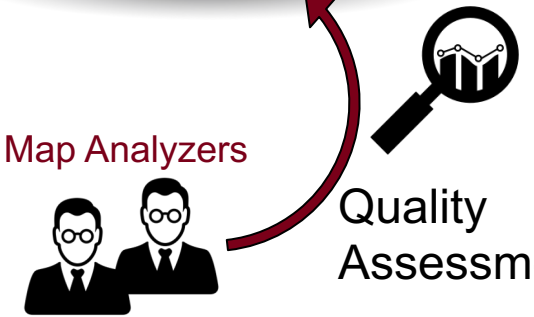




Range Query Routing …

OSM:

- **12+ Billion updates**
- **3 TB Data**

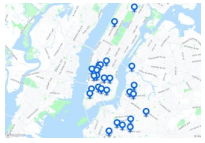


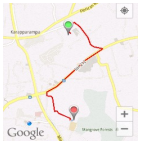
Map Analyzers

Quality Assessment

Not Scalable

Road Network Queries…





Range Query Routing …

OSM:

- **12+ Billion updates**
- **3 TB Data**

Not Scalable

RASED

- Global-Scale
- Comprehensive
- Highly Interactive

How does RASED Help Map Analyzers?

How does RASED Help Map Analyzers?

- Road changes - > gives road network stability
- Query and visualize various statistics
- Numerous query scenarios
- 15 years worth of data
- Scalable, Interactive dashboard

How does RASED Help Map Analyzers?

- Road changes - > gives road network stability
- Query and visualize various statistics
- Numerous query scenarios
- 15 years worth of data
- Scalable, Interactive dashboard

RASED is the first-ever attempt to **quantify** and **visualize all** OSM **worldwide** changes on a **daily** basis

Outline

1. **Motivation**
2. **RASED Queries**
3. **RASED Architecture & Main Modules**
   - Data Collection & Processing
   - Storage & Indexing
   - Query Execution
   - User Interface
4. **Experimental Results**
5. **RASED in Action (A Quick Demo)**

RASED Analysis Queries

Example UpdateList U

Date	Country	RoadType	ElementType	UpdateType	Latitude	Longitude	ChangesetID
06/07/2022	USA	Service	Way	New	44.5	-93.2	5582147
…	…	…	…	…			

RASED Analysis Queries

Example UpdateList U

Date	Country	RoadType	ElementType	UpdateType	Latitude	Longitude	ChangesetID
06/07/2022	USA	Service	Way	New	44.5	-93.2	5582147
…	…	…	…	…			

- **Aggregations**: on any highlighted fields.

RASED Analysis Queries

Example UpdateList U

Date	Country	RoadType	ElementType	UpdateType	Latitude	Longitude	ChangesetID
06/07/2022	USA	Service	Way	New	44.5	-93.2	5582147
…	…	…	…	…			

- **Aggregations**: on any highlighted fields.
- **Filters**: Temporal (e.g., time of update), spatial (e.g., country or state), road types, and update types.

RASED Analysis Queries

Example UpdateList U

Date	Country	RoadType	ElementType	UpdateType	Latitude	Longitude	ChangesetID
06/07/2022	USA	Service	Way	New	44.5	-93.2	5582147
…	…	…	…	…			

- **Aggregations**: on any highlighted fields.
- **Filters**: Temporal (e.g., time of update), spatial (e.g., country or state), road types, and update types.
- **Goal:** find number and percentage of OSM updates:
  - E.g., per country, comparison between countries, types of updated roads, and temporal evolution of updates.

RASED Analysis Queries

RASED Analysis Queries

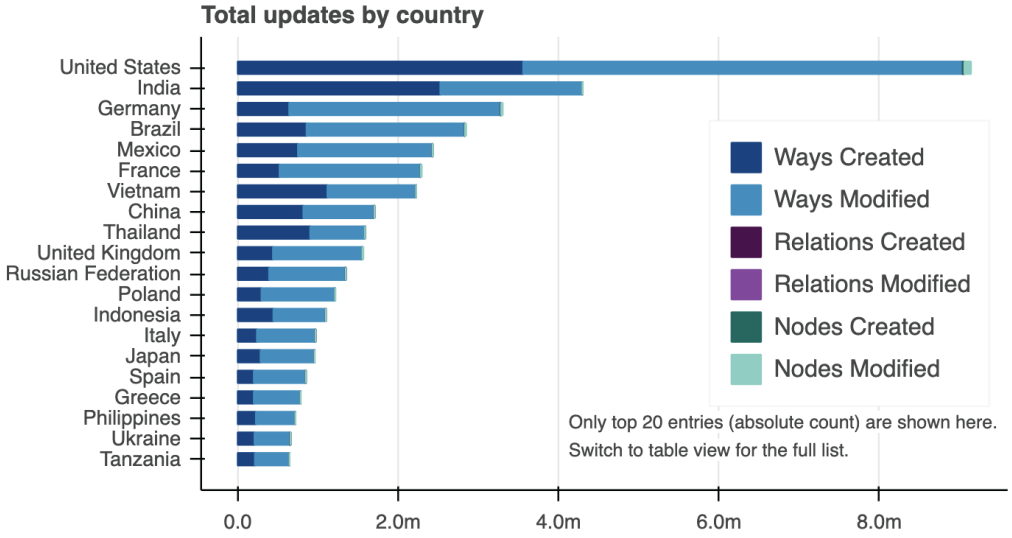
Example 1:

Find the number of newly created or modified element types (node, way, relation) for each country road network in 2021.

RASED Analysis Queries

Example 1:

Find the number of newly created or modified element types (node, way, relation) for each country road network in 2021.

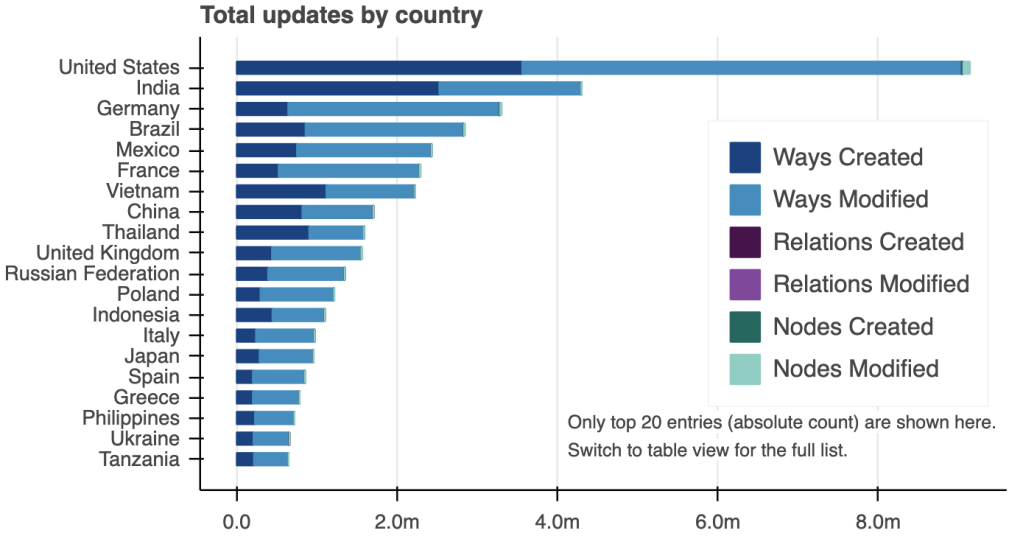


RASED Analysis Queries

Example 1:

Find the number of newly created or modified element types (node, way, relation) for each country road network in 2021.

```
SELECT U.Country,U.ElementType,COUNT(*)
FROM UpdateList U
WHERE U.Date BETWEEN 2021-01-01
  AND 2021-12-31
  AND U.UpdateType IN [New,Update]
GROUP BY U.Country, U.ElementType
```

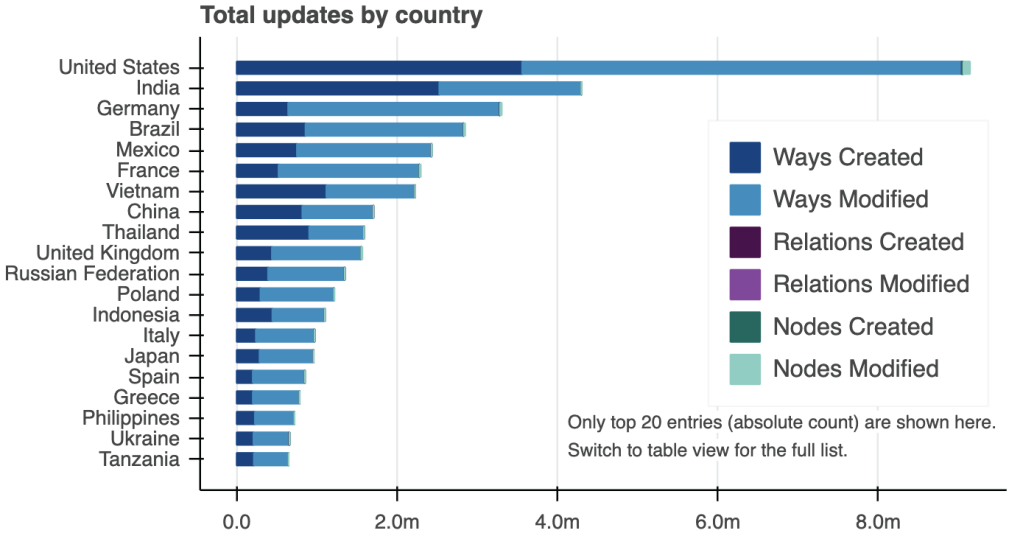


RASED Analysis Queries

Example 1:

Find the number of newly created or modified element types (node, way, relation) for each country road network in 2021.

```
SELECT U.Country,U.ElementType,COUNT(*)
FROM UpdateList U
WHERE U.Date BETWEEN 2021-01-01
  AND 2021-12-31
  AND U.UpdateType IN [New,Update]
GROUP BY U.Country, U.ElementType
```



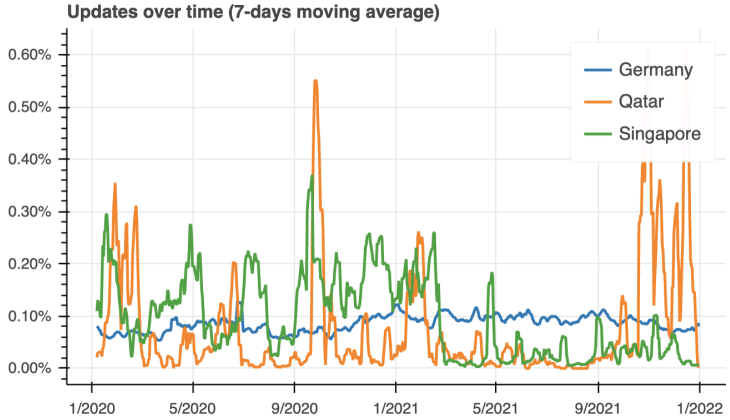
RASED queries are always supported in the order of **milliseconds** regardless of how large is query period.

RASED Analysis Queries

RASED Analysis Queries

Example 2:

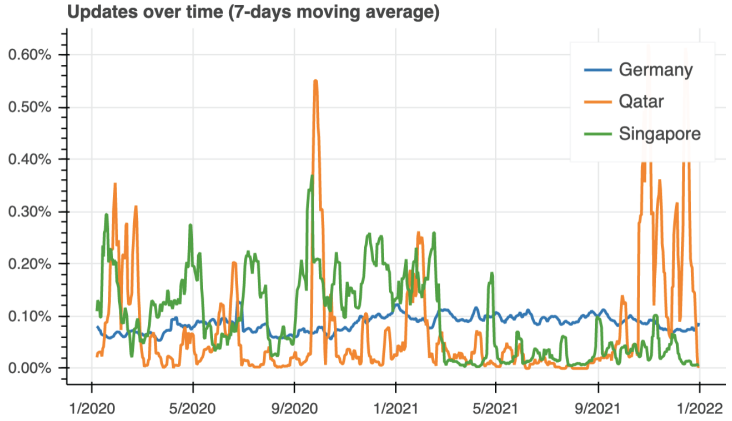
Compare the percentage of daily changes in road network in Germany, Singapore, and Qatar over 2020 and 2021.



RASED Analysis Queries

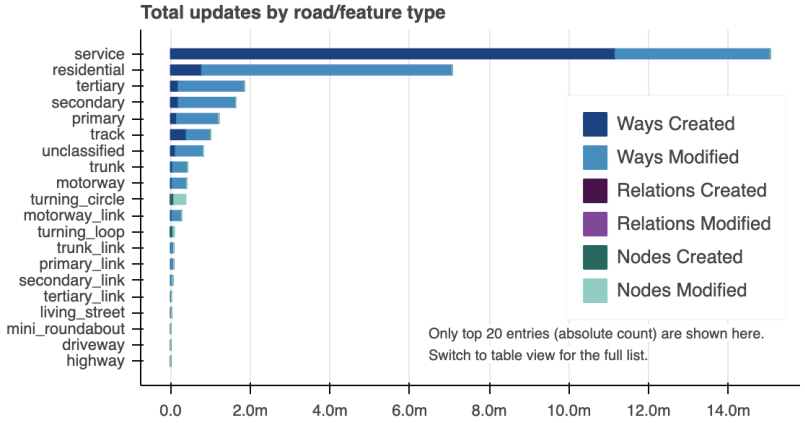
Example 2:

Compare the percentage of daily changes in road network in Germany, Singapore, and Qatar over 2020 and 2021.



Example 3:

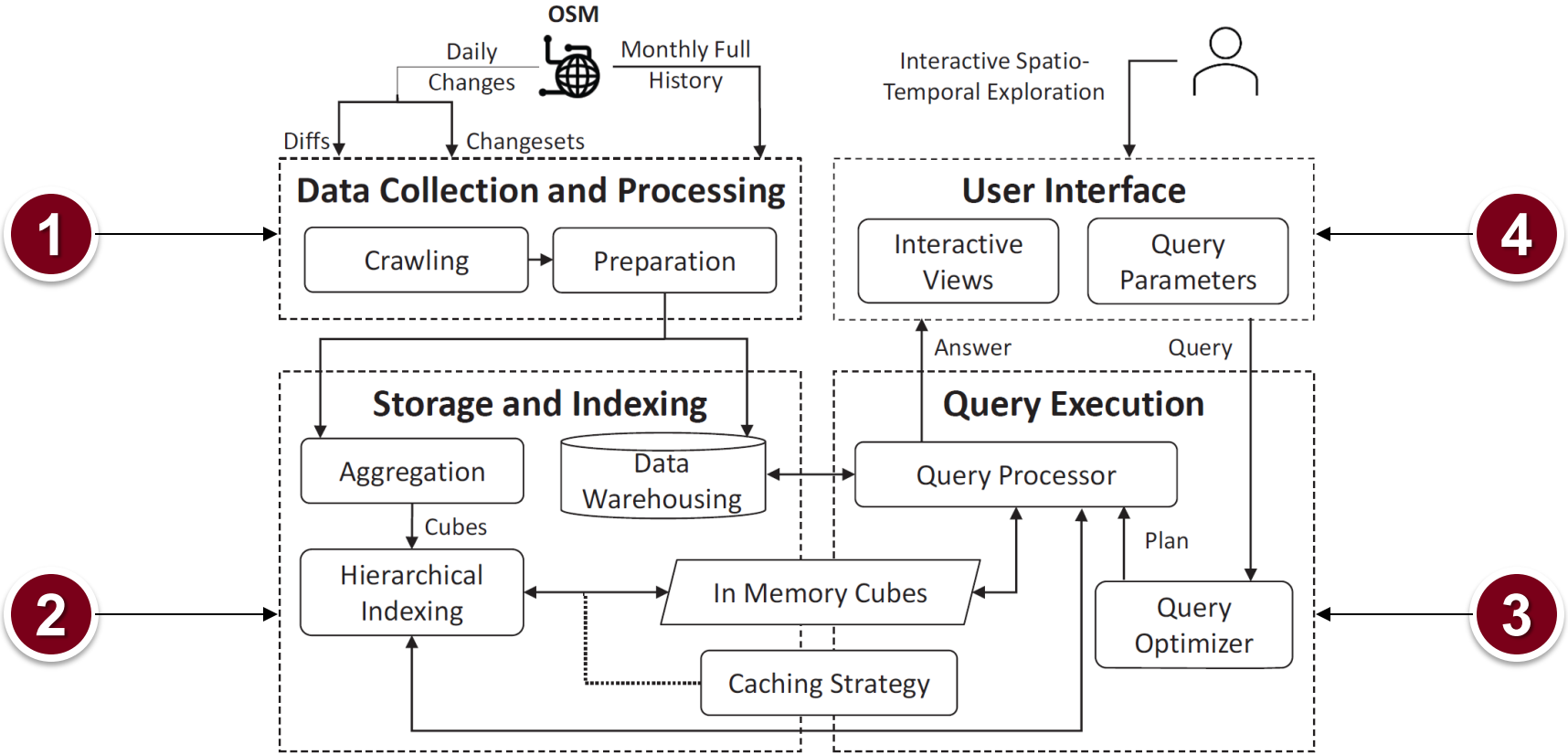
Find the number of newly created or modified elements types (node, way, relation) for each road type in USA since 2018.



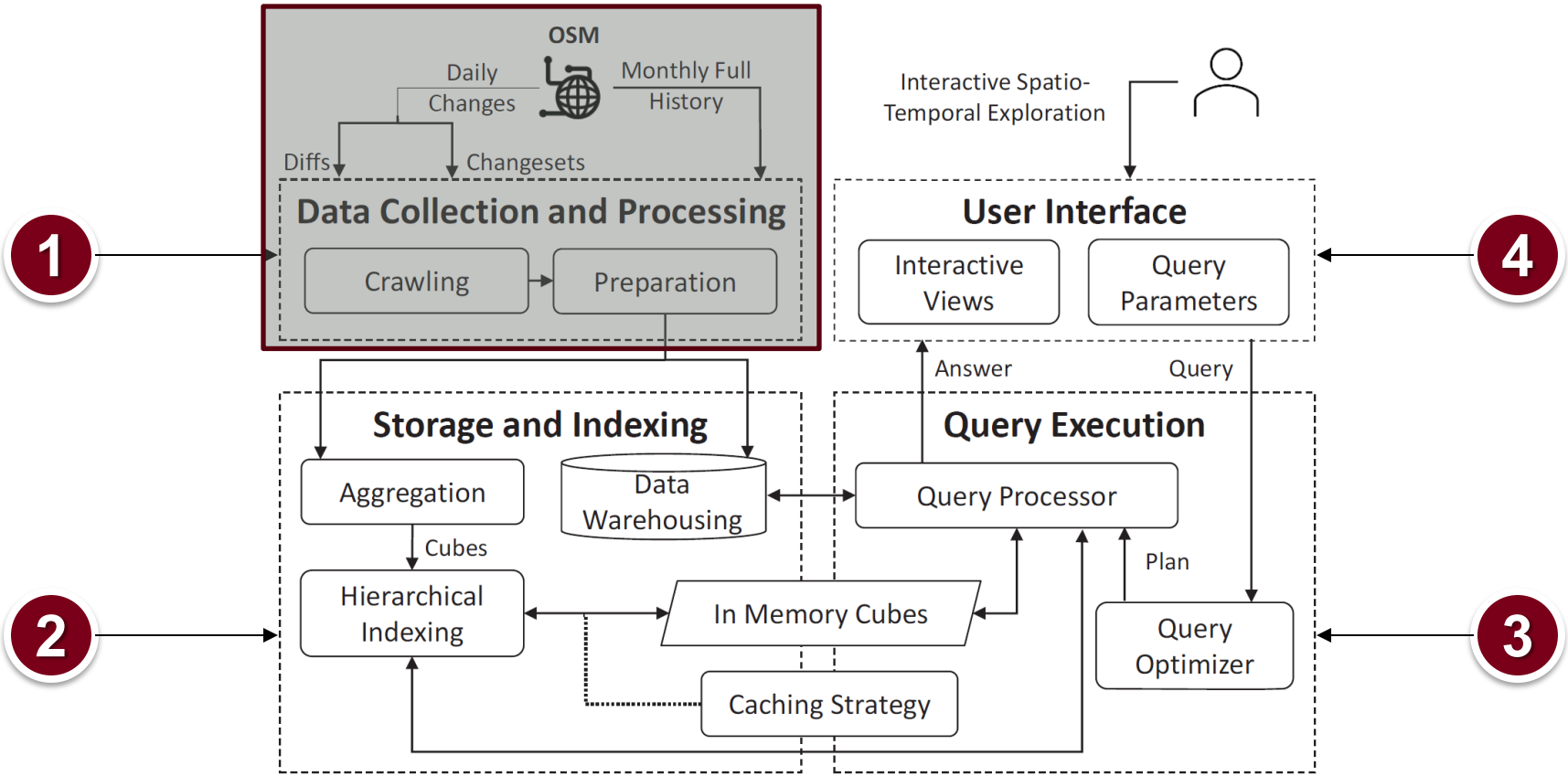
Outline

1. **Motivation**
2. **RASED Queries**
3. **RASED Architecture & Main Modules**
   - Data Collection & Processing
   - Storage & Indexing
   - Query Execution
   - User Interface
4. **Experimental Results**
5. **RASED in Action (A Quick Demo)**

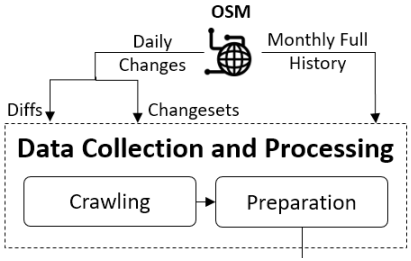
RASED Architecture



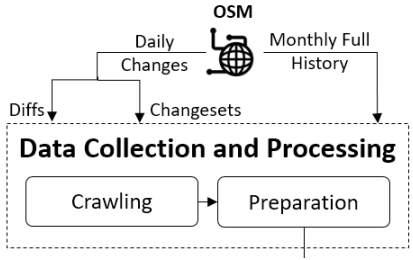
RASED Architecture



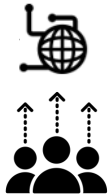
1- Data Collection and Processing



1- Data Collection and Processing

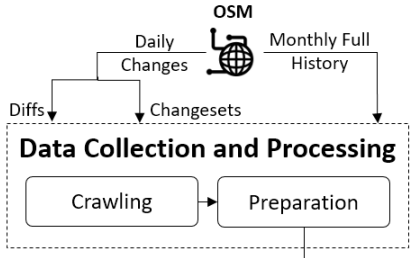


Planet.osm



300K
active users

1- Data Collection and Processing



Planet.osm

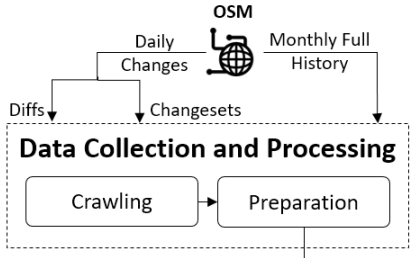
300K active users

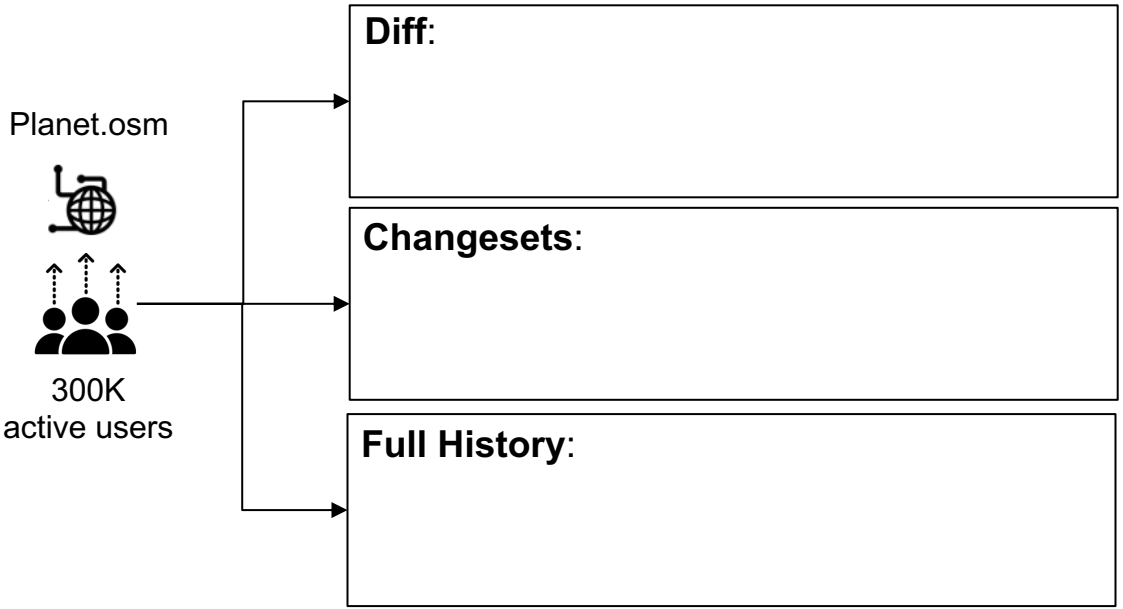
Diff:

Changesets:

Full History:

1- Data Collection and Processing

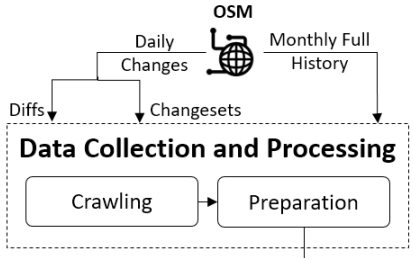




UpdateList U

Date	Country	RoadType	ElementType	UpdateType	Lat	Long	ChangesetID
?	?	?	?	?	?	?	?

1- Data Collection and Processing



Planet.osm

300K active users

Diff: only new versions of updated roads

	Minute	Hour	Day
# Files	5M	82K	3.5K

Few GB each

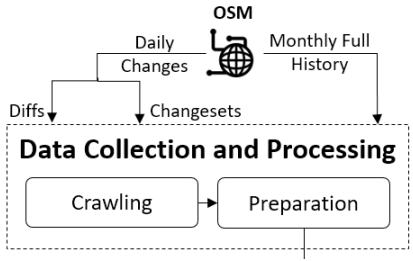
Changesets:

Full History:

UpdateList U

Date	Country	RoadType	ElementType	UpdateType	Lat	Long	ChangesetID
?	?	?	?	?	?	?	?

1- Data Collection and Processing



Planet.osm

300K active users

Diff: only new versions of updated roads

	Minute	Hour	Day
# Files	5M	82K	3.5K

Few GB each

Changesets: updates metadata

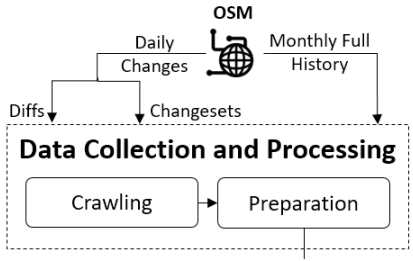
Every 1K Changesets	New file every 1 minute	~ 5M files so far	~50 GB

Full History:

UpdateList U

Date	Country	RoadType	ElementType	UpdateType	Lat	Long	ChangesetID
?	?	?	?	?	?	?	?

1- Data Collection and Processing



Planet.osm

300K active users

Diff: only new versions of updated roads

	Minute	Hour	Day
# Files	5M	82K	3.5K

Few GB each

Changesets: updates metadata

Every 1K Changesets	New file every 1 minute	~ 5M files so far	~50 GB

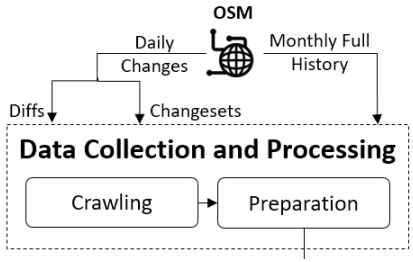
Full History:

Daily Crawler

UpdateList U

Date	Country	RoadType	ElementType	UpdateType	Lat	Long	ChangesetID
06/07/2022	USA	Service	Way	?	44.5	-93.2	5582147
…	…	…	…				

1- Data Collection and Processing



Planet.osm

300K active users

Diff: only new versions of updated roads

	Minute	Hour	Day
# Files	5M	82K	3.5K

Few GB each

Changesets: updates metadata

Every 1K Changesets	New file every 1 minute	~ 5M files so far	~50 GB

Full History: previous state of each update

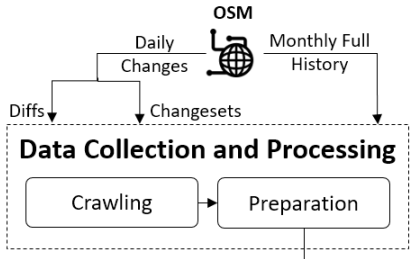
~ every month	12+ Billion updates/versions	3 TB

Daily Crawler

UpdateList U

Date	Country	RoadType	ElementType	UpdateType	Lat	Long	ChangesetID
06/07/2022	USA	Service	Way	?	44.5	-93.2	5582147
…	…	…	…				

1- Data Collection and Processing



Planet.osm

300K active users

Diff: only new versions of updated roads

	Minute	Hour	Day
# Files	5M	82K	3.5K

Few GB each

Changesets: updates metadata

Every 1K Changesets	New file every 1 minute	~ 5M files so far	~50 GB

Full History: previous state of each update

~ every month	12+ Billion updates/versions	3 TB

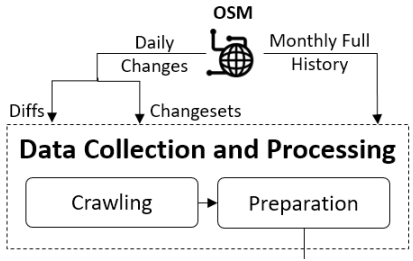
Daily Crawler

Monthly Crawler

UpdateList U

Date	Country	RoadType	ElementType	UpdateType	Lat	Long	ChangesetID
06/07/2022	USA	Service	Way	Modified Geometry	44.5	-93.2	5582147
…	…	…	…				

1- Data Collection and Processing



Planet.osm

300K active users

Diff: only new versions of updated roads

	Minute	Hour	Day
# Files	5M	82K	3.5K

Few GB each

Changesets: updates metadata

Every 1K Changesets	New file every 1 minute	~ 5M files so far	~50 GB

Full History: previous state of each update

~ every month	12+ Billion updates/versions	3 TB

Up to date statistics

Daily Crawler

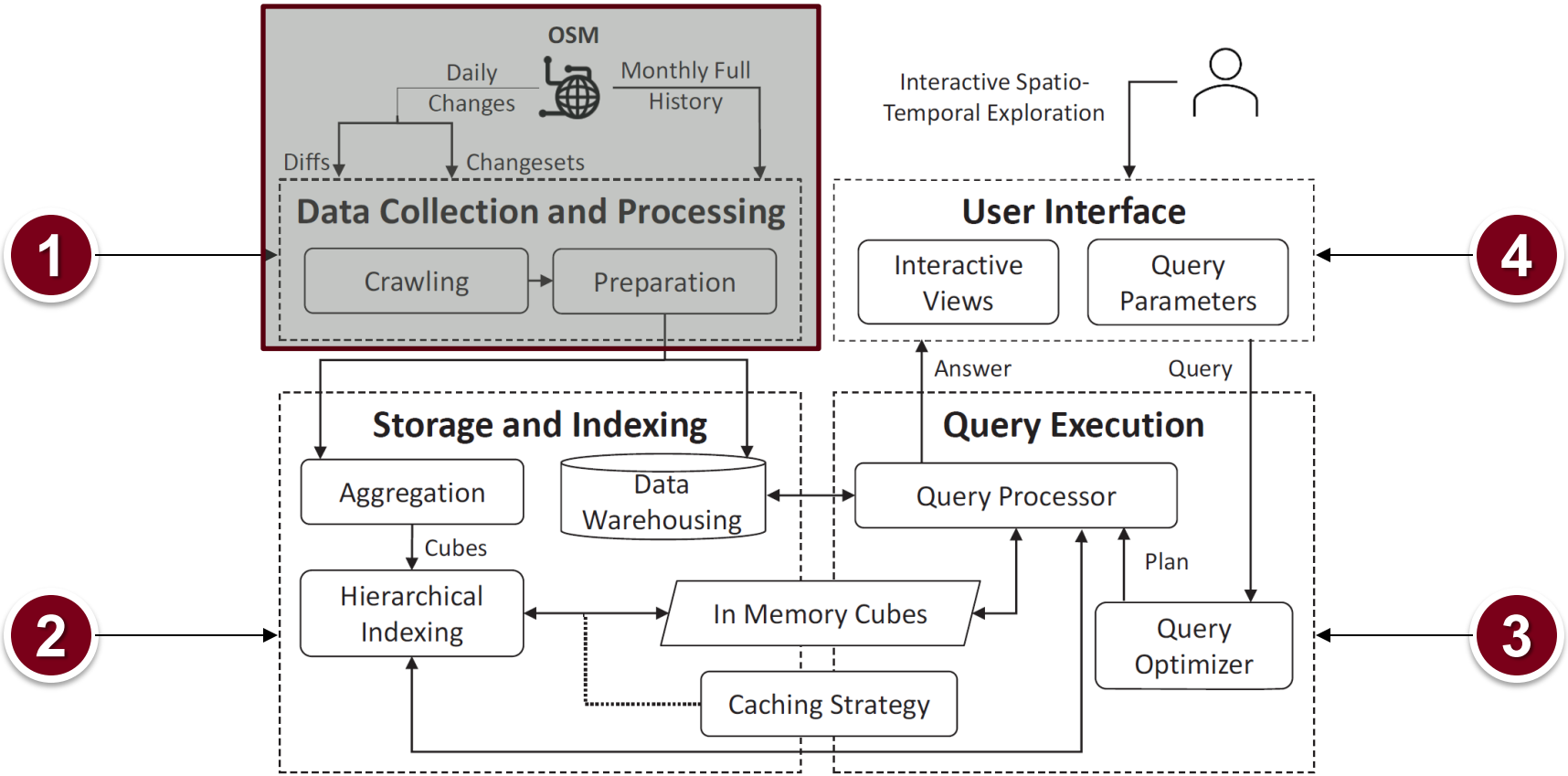
More Details
Complete Missing information

Monthly Crawler

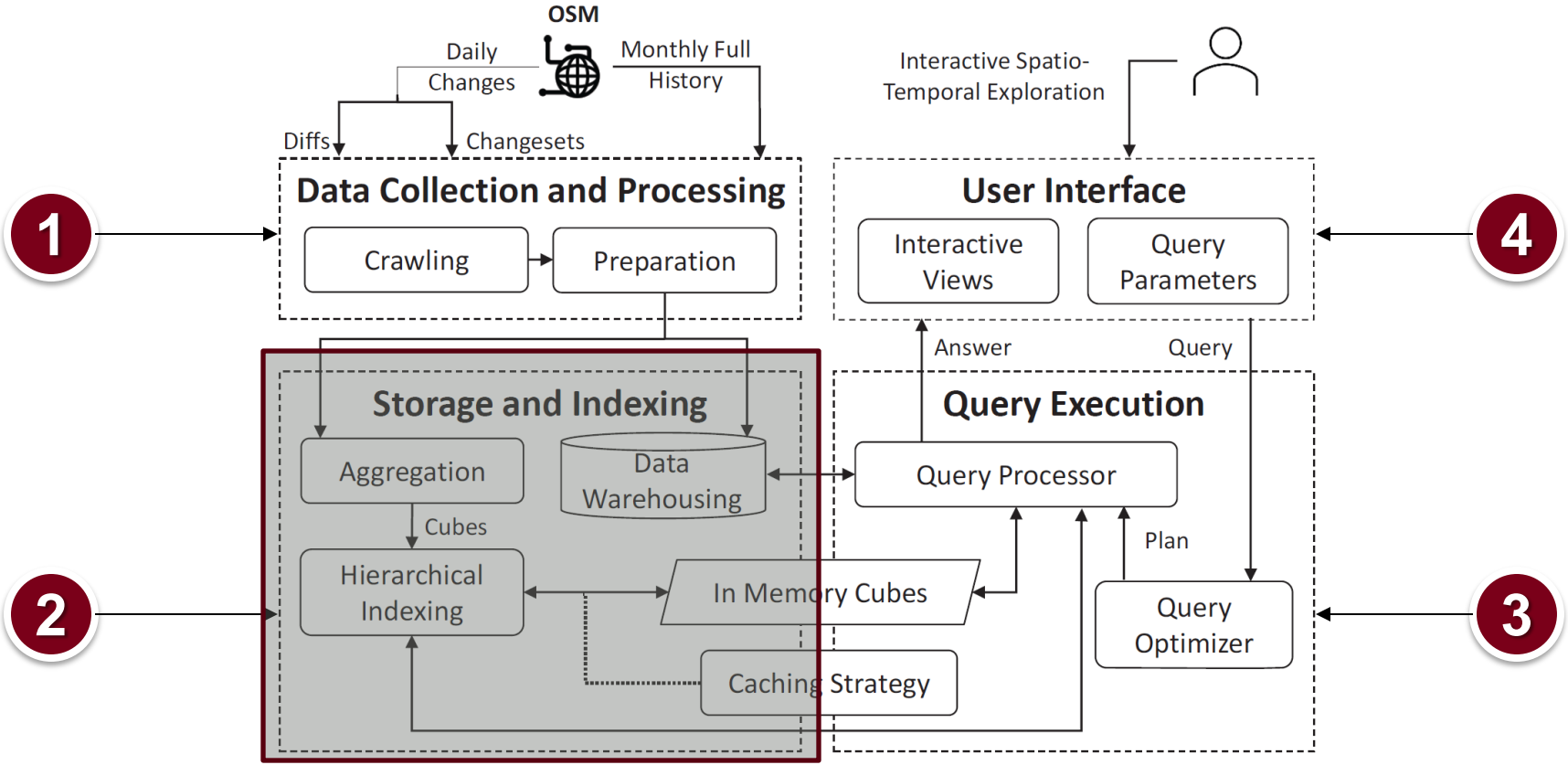
UpdateList U

Date	Country	RoadType	ElementType	UpdateType	Lat	Long	ChangesetID
06/07/2022	USA	Service	Way	Modified Geometry	44.5	-93.2	5582147
…	…	…	…				

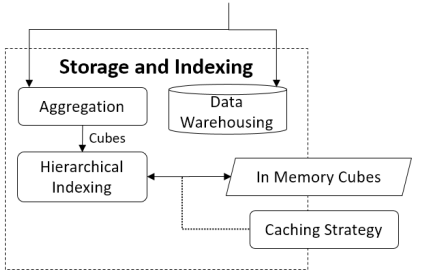
RASED Architecture



RASED Architecture

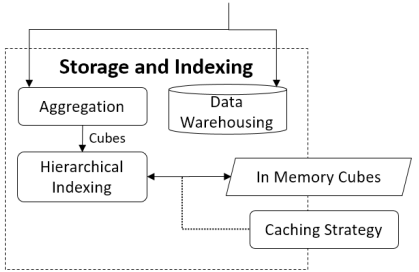


2- Storage and Indexing

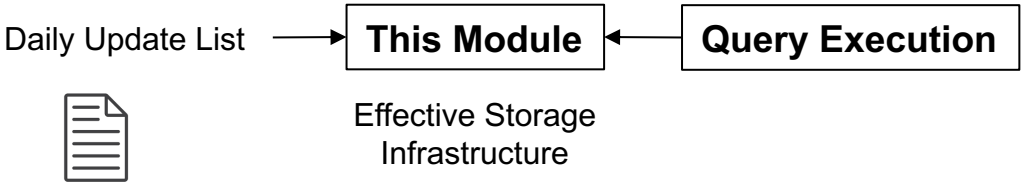


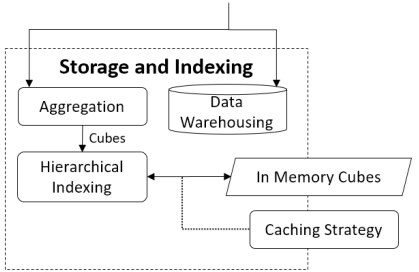
2- Storage and Indexing

Daily Update List → **This Module**



2- Storage and Indexing





2- Storage and Indexing

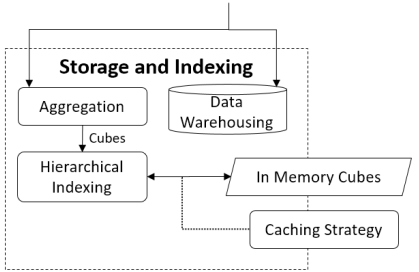
Daily Update List → **This Module** ← **Query Execution**

Effective Storage Infrastructure

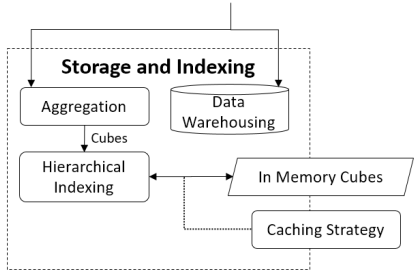
Very few I/O

RASED Analytical Queries:

- Global-Scale
- Comprehensive
- Highly Interactive



2- Storage and Indexing



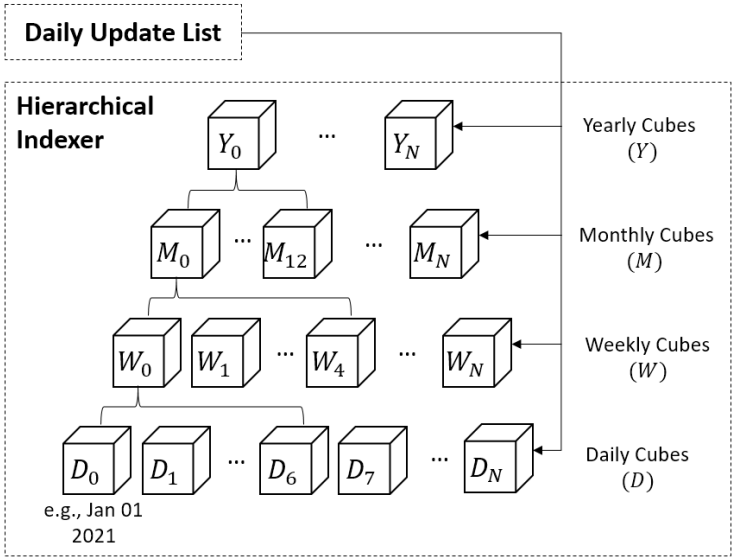
Daily Update List → **This Module** ← **Query Execution**

Effective Storage Infrastructure

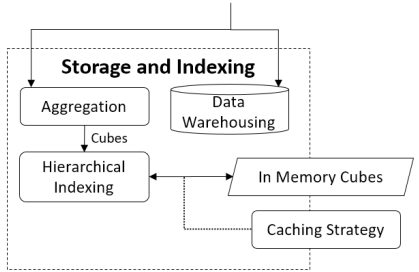
Very few I/O

RASED Analytical Queries:
- Global-Scale
- Comprehensive
- Highly Interactive

Hierarchal Temporal Index



2- Storage and Indexing



Daily Update List → **This Module** ← **Query Execution**

Effective Storage Infrastructure

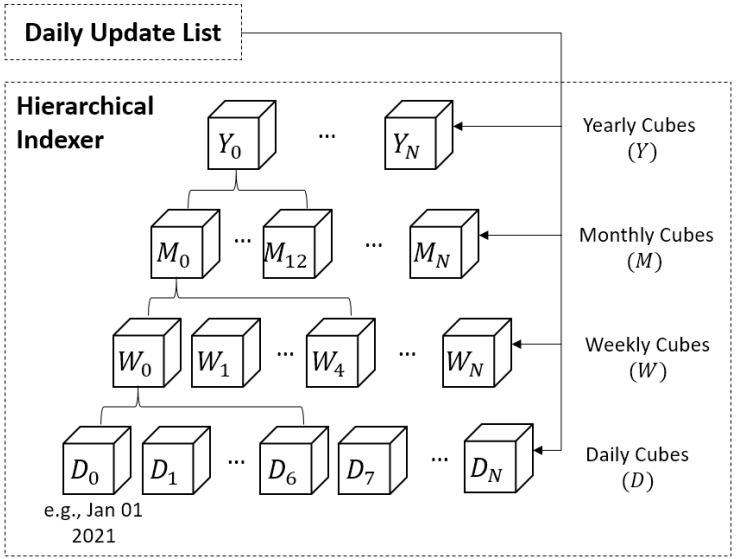
Very few I/O

RASED Analytical Queries:
- Global-Scale
- Comprehensive
- Highly Interactive

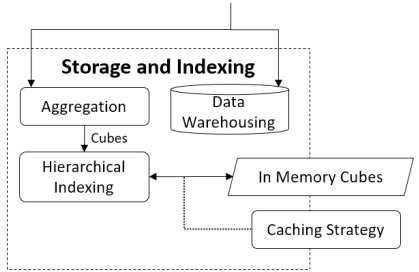
Index Nodes

- Precomputed statistics, not map updates itself
- Stored as 4-dimential data cubes:
  - ***Country***: 300+ values
  - **Road Type**: 150 values
  - **Update Type**: 4 values
  - **Element Type**: 3 values.

Hierarchal Temporal Index



2- Storage and Indexing



Daily Update List → **This Module** ← **Query Execution**

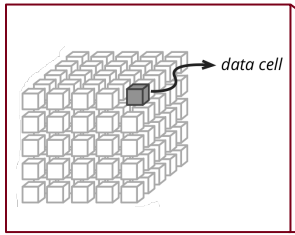
Effective Storage Infrastructure

Very few I/O

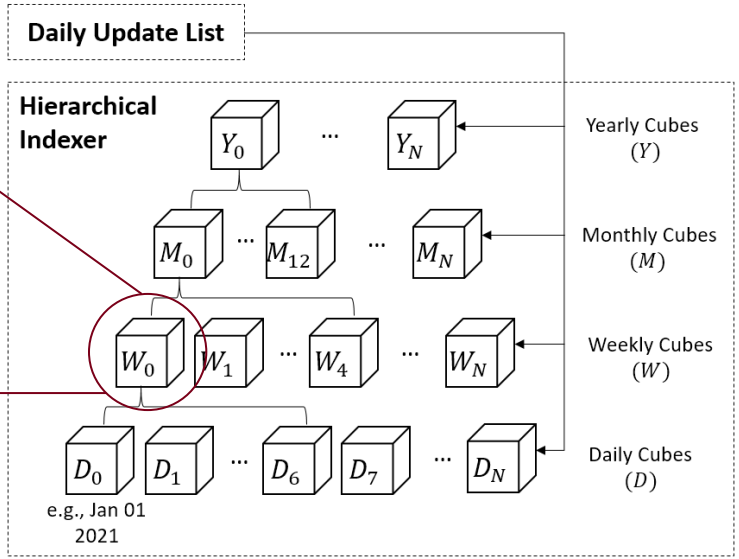
RASED Analytical Queries:
- Global-Scale
- Comprehensive
- Highly Interactive

Index Nodes

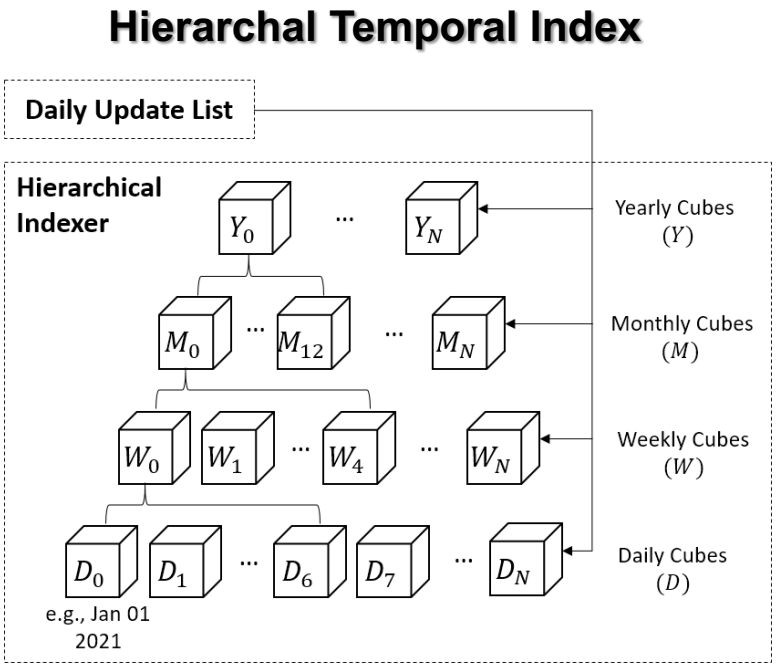
- Precomputed statistics, not map updates itself
- Stored as 4-dimential data cubes:
  - ***Country***: 300+ values
  - **Road Type**: 150 values
  - **Update Type**: 4 values
  - **Element Type**: 3 values.
- Each cube:
  - has 540,000 precomputed values (cells)
  - covers a specific period (e.g., week)
  - ~4MB, one disk page.
- Each cube cell is the aggregate count of OSM updates



Hierarchal Temporal Index



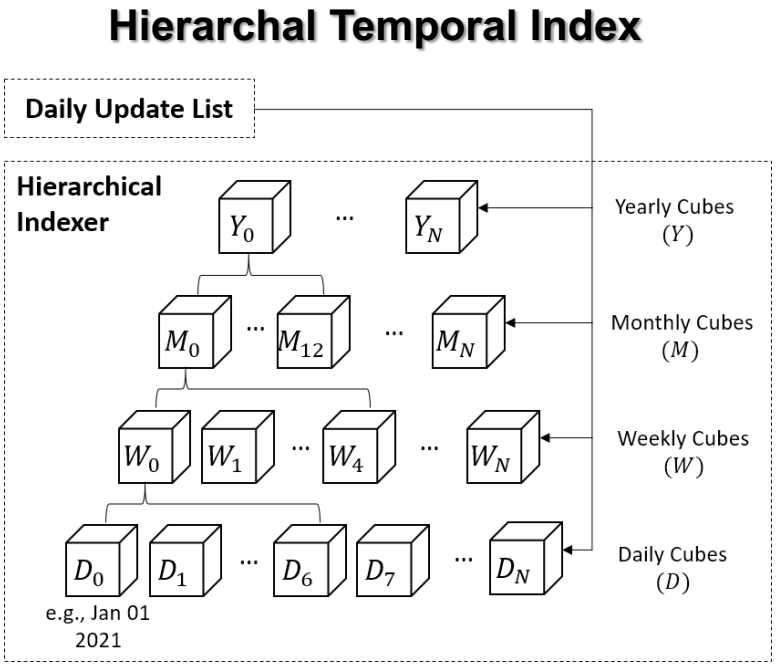
RASED Hierarchical Temporal Indexing



RASED Hierarchical Temporal Indexing

Index Hierarchy

Four levels:
- yearly
- monthly
- weekly
- daily



RASED Hierarchical Temporal Indexing

Index Hierarchy

Four levels:
- yearly
- monthly
- weekly
- daily

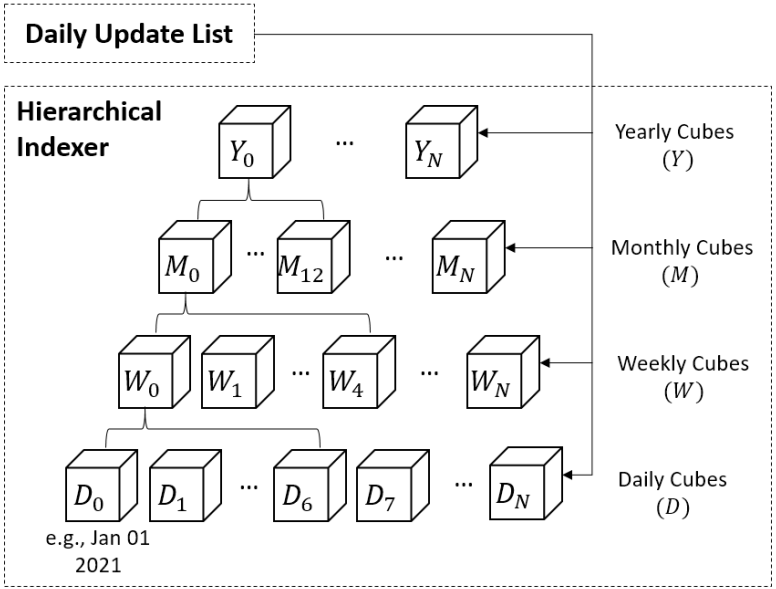
Index Size

OSM updates since 2004:

6,000+ daily nodes
200+ monthly nodes
850+ weekly nodes
16 yearly nodes

7,000 nodes / **4 billion** aggregate values / **~28GB**

Hierarchal Temporal Index



RASED Hierarchical Temporal Indexing

Index Hierarchy

Four levels:
- yearly
- monthly
- weekly
- daily

Index Size

OSM updates since 2004:

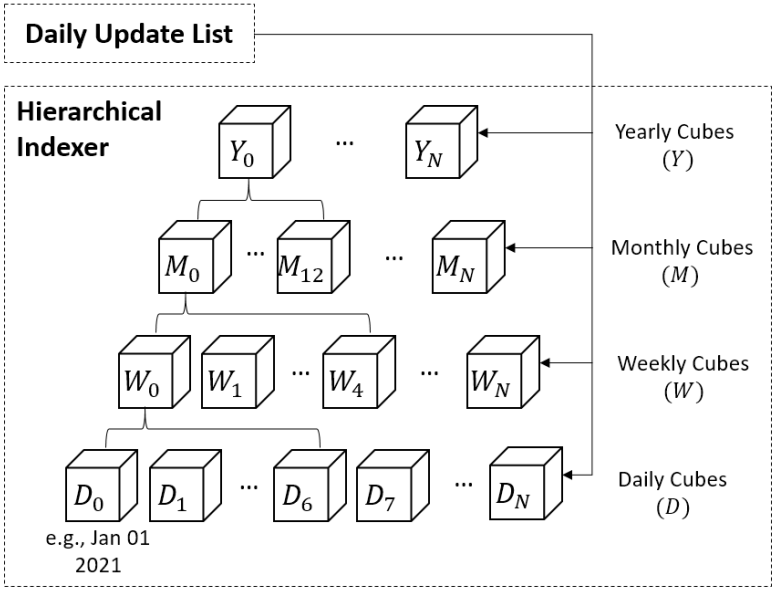
6,000+ daily nodes
200+ monthly nodes
850+ weekly nodes
16 yearly nodes

7,000 nodes / **4 billion** aggregate values / **~28GB**

Index Maintenance

Daily updates :
- every day constructs & appends new daily cube
- Recursively build weekly/monthly/yearly cubes
- offline, ~30 minutes

Hierarchal Temporal Index



RASED Hierarchical Temporal Indexing

Index Hierarchy

Four levels:
- yearly
- monthly
- weekly
- daily

Index Size

OSM updates since 2004:

6,000+ daily nodes
850+ weekly nodes
200+ monthly nodes
16 yearly nodes

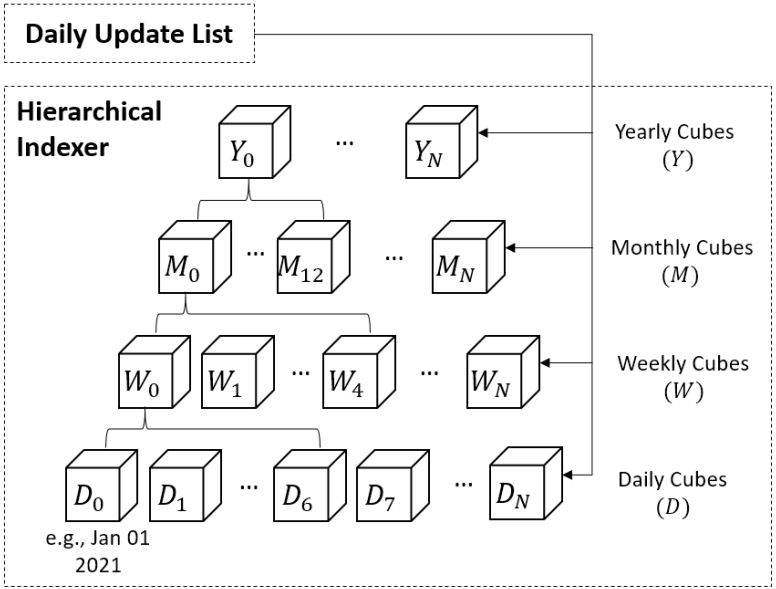
7,000 nodes / **4 billion** aggregate values / **~28GB**

Index Maintenance

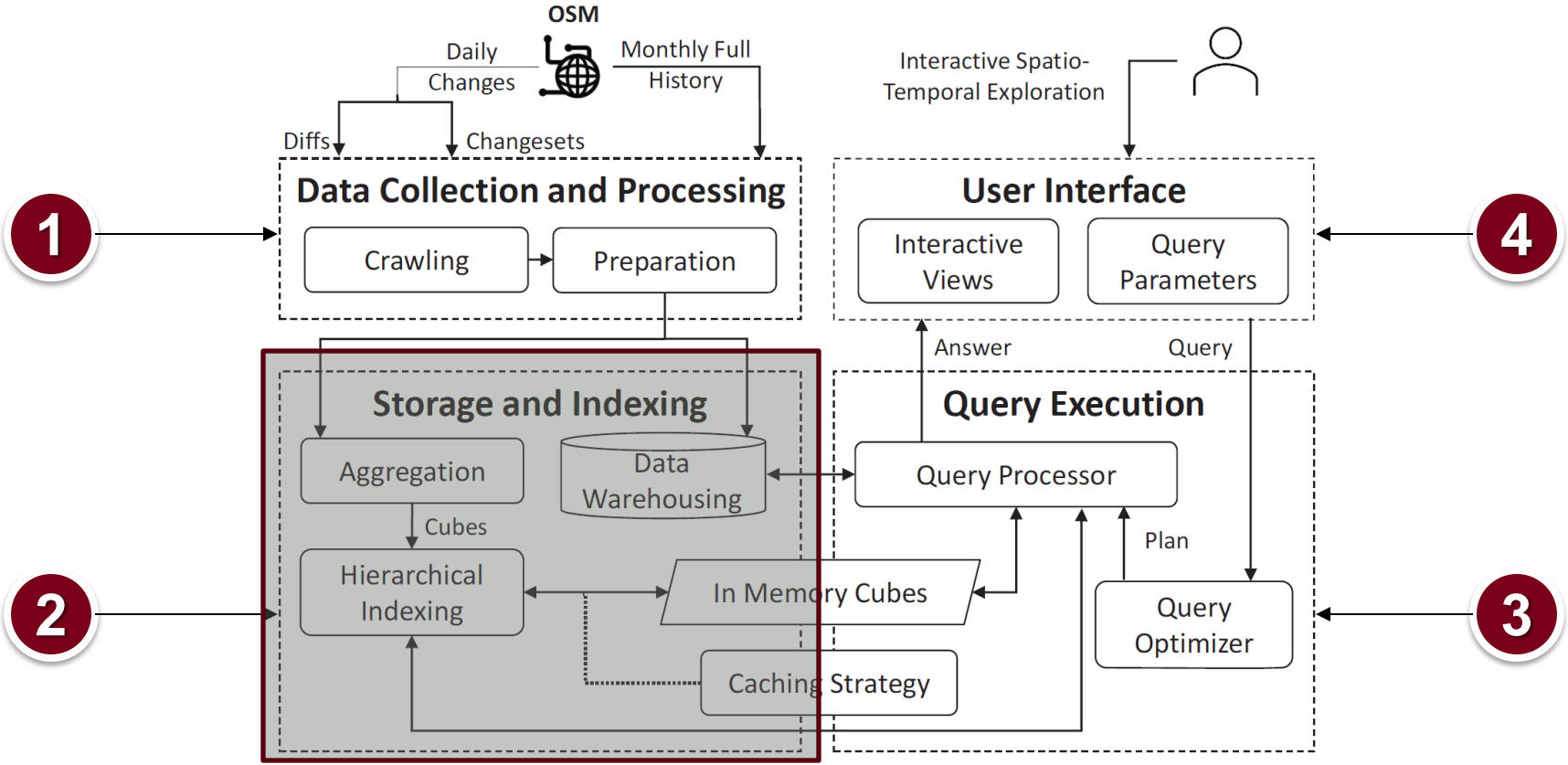
Daily updates :
- every day constructs & appends new daily cube
- Recursively build weekly/monthly/yearly cubes
- offline, ~30 minutes

Monthly updates: more information obtained:
- reconstruct daily & weekly cubes
- offline, ~few hours.

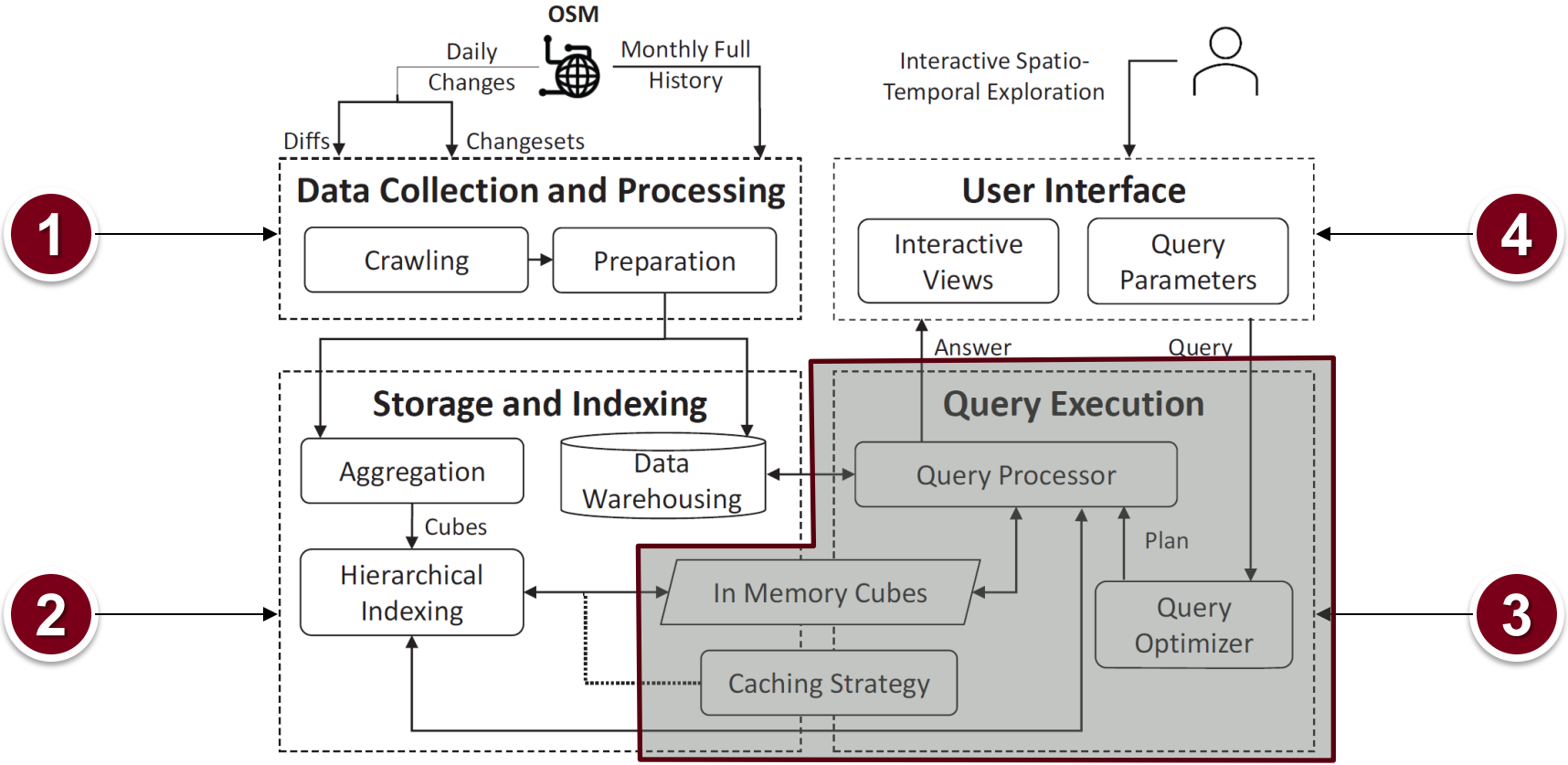
Hierarchal Temporal Index



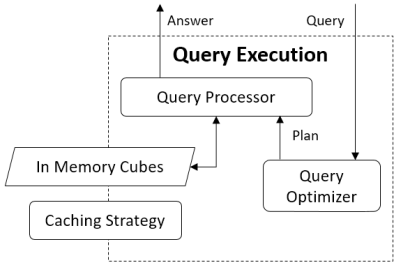
RASED Architecture



RASED Architecture

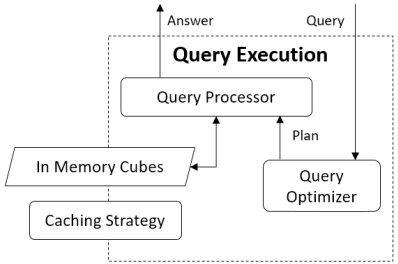


3- Query Execution

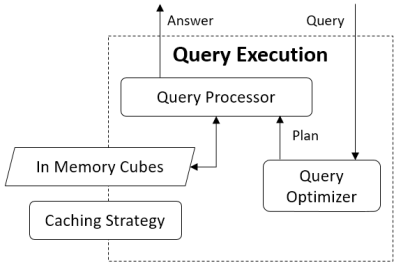


3- Query Execution

Ex. Find number of updates in each country in time window t.



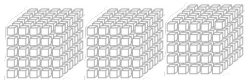
3- Query Execution



Ex. Find number of updates in each country in time window t.

Two phases:

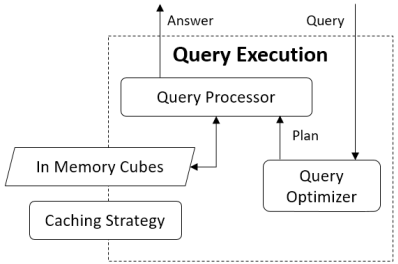
Retrieves cubes satisfies t



All dimensions

1

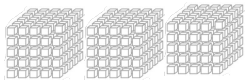
3- Query Execution



Ex. Find number of updates in each country in time window t.

Two phases:

Retrieves cubes satisfies t

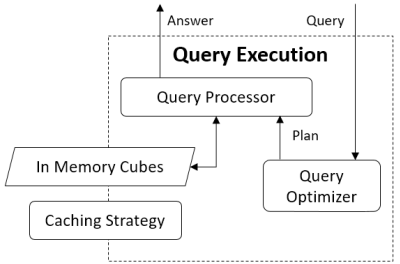


All dimensions

1

Disk-Based

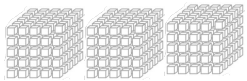
3- Query Execution



Ex. Find number of updates in each country in time window t.

Two phases:

Retrieves cubes satisfies t



All dimensions

1

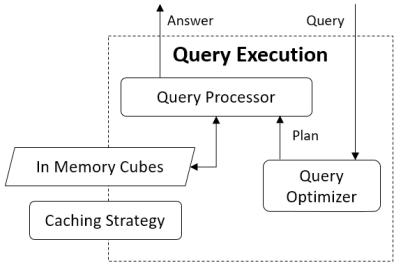
Disk-Based

Aggregates unwanted dimensions
Keep only requested ones

Country

2

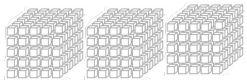
3- Query Execution



Ex. Find number of updates in each country in time window t.

Two phases:

Retrieves cubes satisfies t



All dimensions

1

Disk-Based

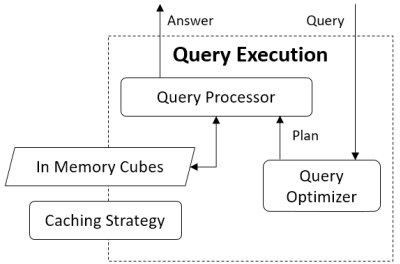
Aggregates unwanted dimensions
Keep only requested ones

Country

2

In-Memory

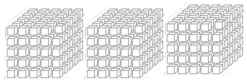
3- Query Execution



Ex. Find number of updates in each country in time window t.

Two phases:

Retrieves cubes satisfies t



All dimensions

1

Disk-Based

Aggregates unwanted dimensions
Keep only requested ones

Country

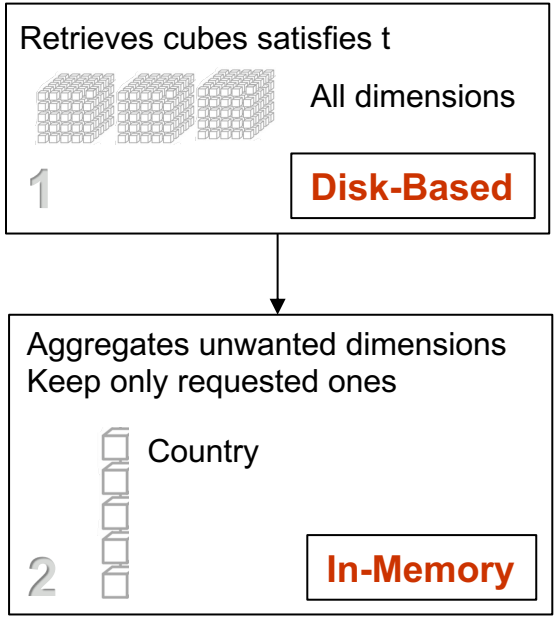
2

In-Memory

3- Query Execution

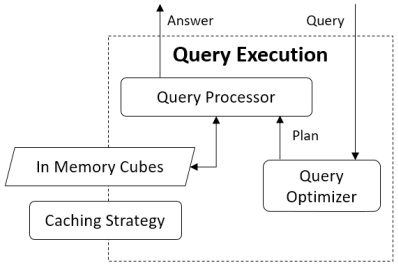
Ex. Find number of updates in each country in time window t.

Two phases:

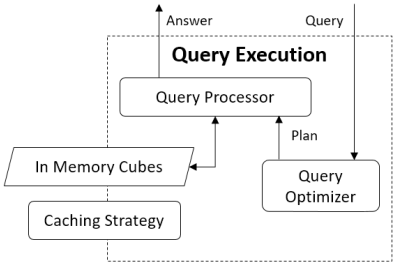


Two Disk Access optimization techniques

Caching & Level Optimization



3- Query Execution

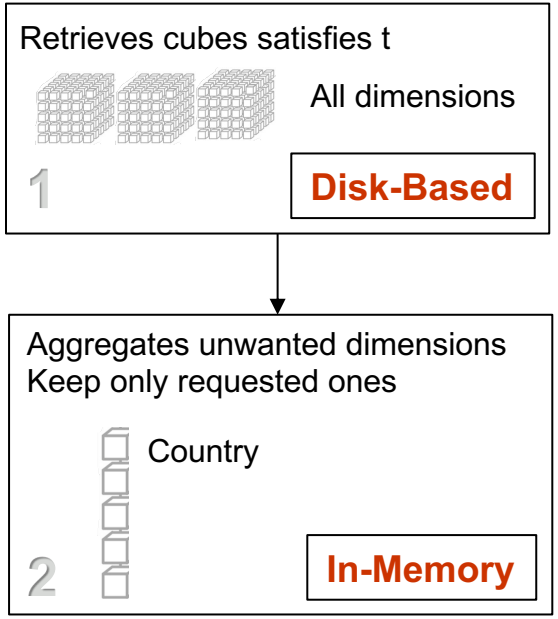


Ex. Find number of updates in each country in time window t.

Two Disk Access optimization techniques

Caching & Level Optimization

Two phases:

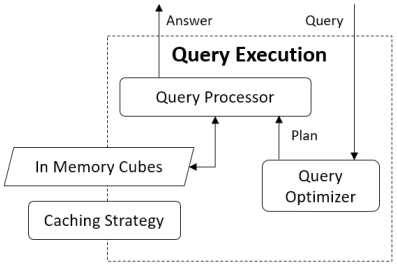


A. Caching

Goal:

- Preload recent data cubes into memory
- Reduce cubes retrieved from disk
- Reduce query response time

3- Query Execution

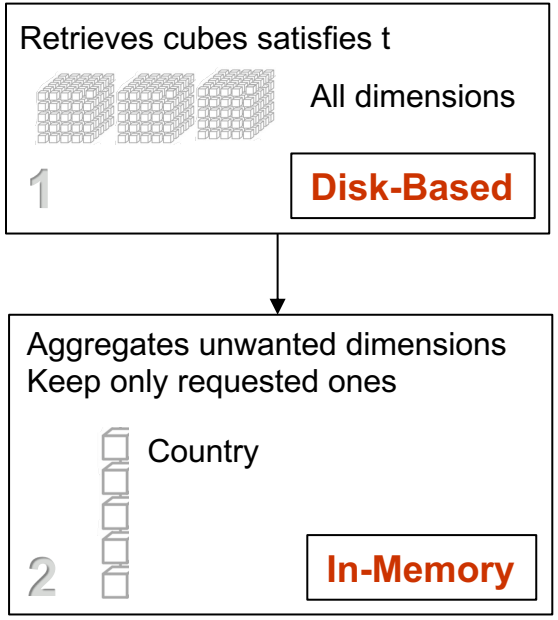


Ex. Find number of updates in each country in time window t.

Two Disk Access optimization techniques

Caching & Level Optimization

Two phases:



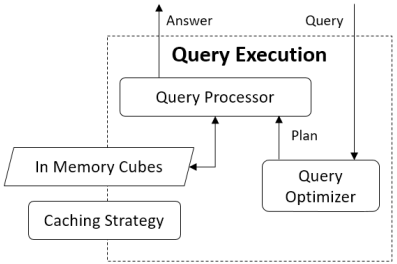
A. Caching

Goal:
- Preload recent data cubes into memory
- Reduce cubes retrieved from disk
- Reduce query response time

Rational:
More likely to receive queries for recent updates

Challenge:
which level to preloaded data cubes from?

3- Query Execution



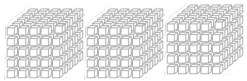
Ex. Find number of updates in each country in time window t.

Two Disk Access optimization techniques

Caching & Level Optimization

Two phases:

Retrieves cubes satisfies t



All dimensions

1

Disk-Based

Aggregates unwanted dimensions
Keep only requested ones

Country

2

In-Memory

A. Caching

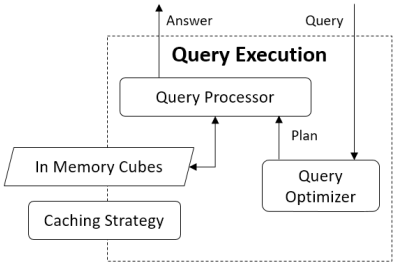
Goal:
- Preload recent data cubes into memory
- Reduce cubes retrieved from disk
- Reduce query response time

Rational:
More likely to receive queries for recent updates

Challenge:
which level to preloaded data cubes from?

$$\{D_{|D|-i}\}_{i=0}^{\alpha N} \cup \{W_{|W|-i}\}_{i=0}^{\beta N} \cup \{M_{|M|-i}\}_{i=0}^{\gamma N} \cup \{Y_{|Y|-i}\}_{i=0}^{\theta N}$$

3- Query Execution



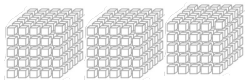
Ex. Find number of updates in each country in time window t.

Two Disk Access optimization techniques

Caching & Level Optimization

Two phases:

Retrieves cubes satisfies t



All dimensions

1 **Disk-Based**

Aggregates unwanted dimensions
Keep only requested ones

Country

2 **In-Memory**

A. Caching

Goal:
- Preload recent data cubes into memory
- Reduce cubes retrieved from disk
- Reduce query response time

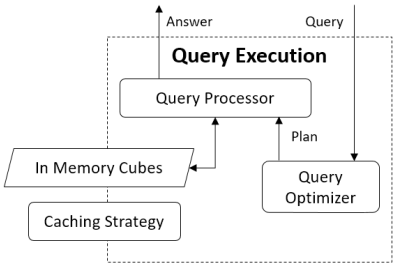
Rational:
More likely to receive queries for recent updates

Challenge:
which level to preloaded data cubes from?

$$\{D_{|D|-i}\}_{i=0}^{\alpha N} \cup \{W_{|W|-i}\}_{i=0}^{\beta N} \cup \{M_{|M|-i}\}_{i=0}^{\gamma N} \cup \{Y_{|Y|-i}\}_{i=0}^{\theta N}$$

- N available memory
- α, β, γ, and θ: ratio allocated to each level
- trade off: *aggregation granularity* vs *time coverage*

3- Query Execution



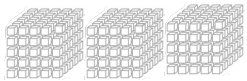
Ex. Find number of updates in each country in time window t.

Two Disk Access optimization techniques

Caching & Level Optimization

Two phases:

Retrieves cubes satisfies t



All dimensions

1

Disk-Based

Aggregates unwanted dimensions
Keep only requested ones

Country

2

In-Memory

A. Caching

Goal:
- Preload recent data cubes into memory
- Reduce cubes retrieved from disk
- Reduce query response time

Rational:
More likely to receive queries for recent updates

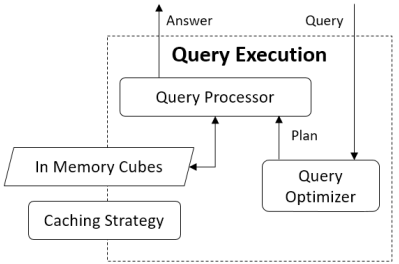
Challenge:
which level to preloaded data cubes from?

$$\{D_{|D|-i}\}_{i=0}^{\alpha N} \cup \{W_{|W|-i}\}_{i=0}^{\beta N} \cup \{M_{|M|-i}\}_{i=0}^{\gamma N} \cup \{Y_{|Y|-i}\}_{i=0}^{\theta N}$$

Days Weeks Months Years

- N available memory
- α, β, γ, and θ: ratio allocated to each level
- trade off: *aggregation granularity* vs *time coverage*

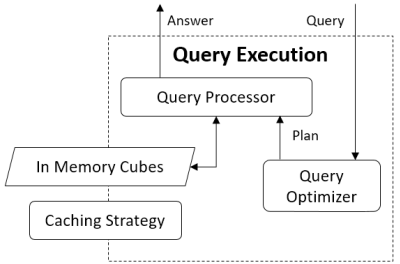
3- Query Execution



Two Disk Access optimization techniques

Caching & Level Optimization

3- Query Execution

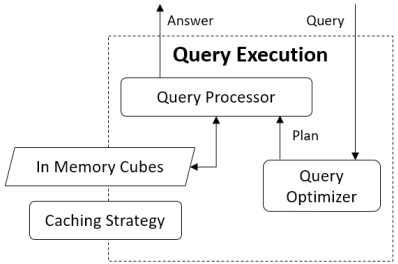


Two Disk Access optimization techniques

Caching & Level Optimization

B. Level Optimization

3- Query Execution



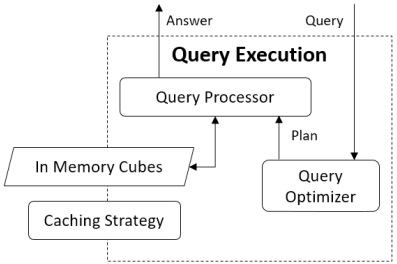
Two Disk Access optimization techniques

Caching & Level Optimization

Example:
Find number of updates in during the period Jan 1 to Feb 15.
(46 days)

B. Level Optimization

3- Query Execution

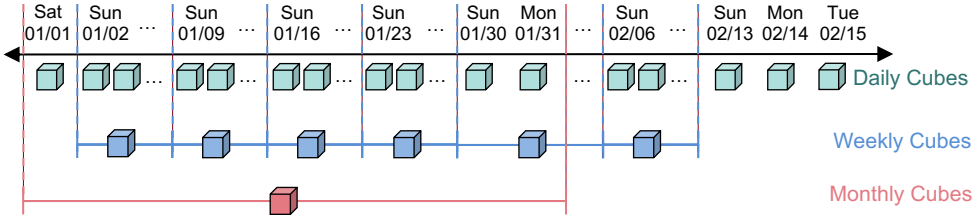


Two Disk Access optimization techniques

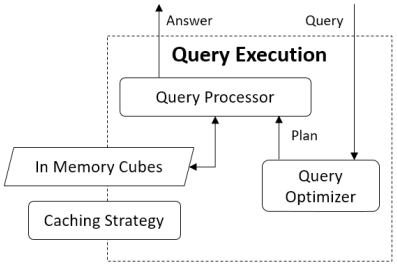
Caching & Level Optimization

Example:
Find number of updates in during the period Jan 1 to Feb 15. (46 days)

B. Level Optimization



3- Query Execution

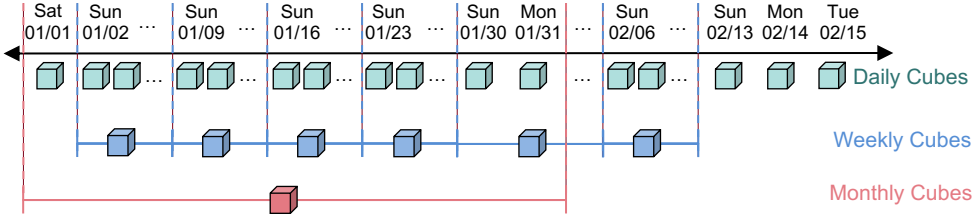


Two Disk Access optimization techniques

Caching & Level Optimization

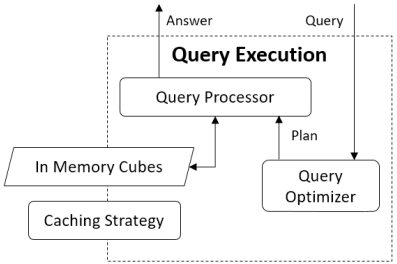
Example:
Find number of updates in during the period Jan 1 to Feb 15.
(46 days)

B. Level Optimization



Three Possible Paths:

3- Query Execution

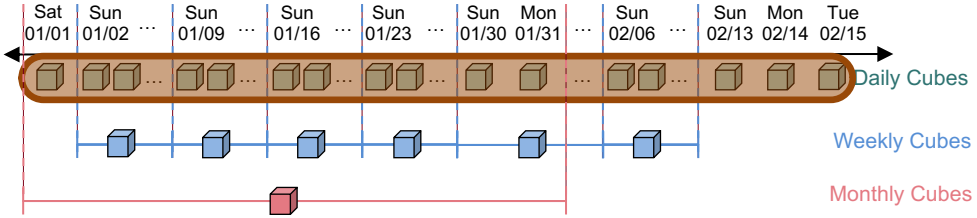


Two Disk Access optimization techniques

Caching & Level Optimization

Example:
Find number of updates in during the period Jan 1 to Feb 15.
(46 days)

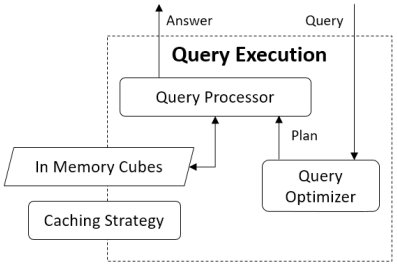
B. Level Optimization



Three Possible Paths:

A- All daily cost = **46 cubes**

3- Query Execution

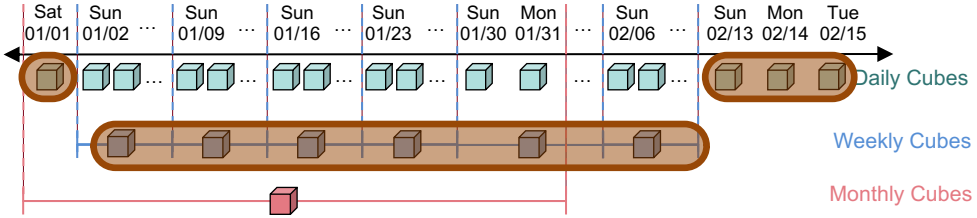


Two Disk Access optimization techniques

Caching & Level Optimization

Example:
Find number of updates in during the period Jan 1 to Feb 15.
(46 days)

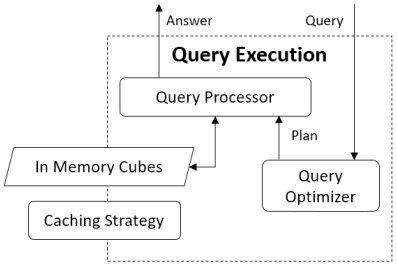
B. Level Optimization



Three Possible Paths:

A- All daily — cost = **46 cubes**
B- 6 weekly + 4 daily — cost = **10 cubes**

3- Query Execution

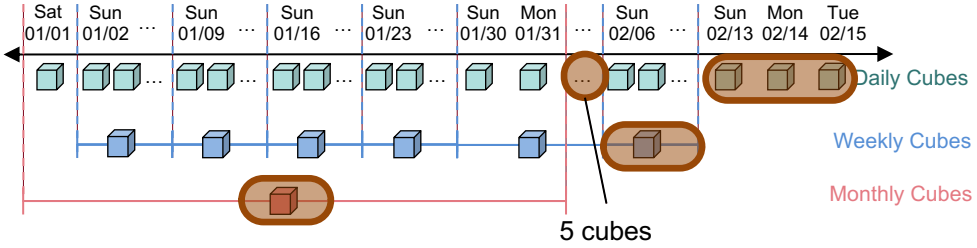


Two Disk Access optimization techniques

Caching & Level Optimization

Example:
Find number of updates in during the period Jan 1 to Feb 15.
(46 days)

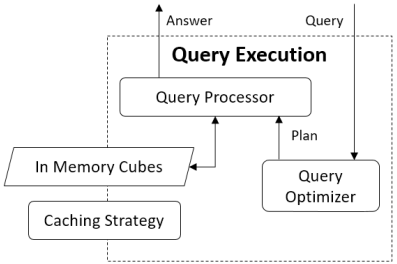
B. Level Optimization



Three Possible Paths:

A- All daily — cost = **46 cubes**
B- 6 weekly + 4 daily — cost = **10 cubes**
C- 1 monthly + 1 weekly + 8 daily — cost = **10 cubes**

3- Query Execution

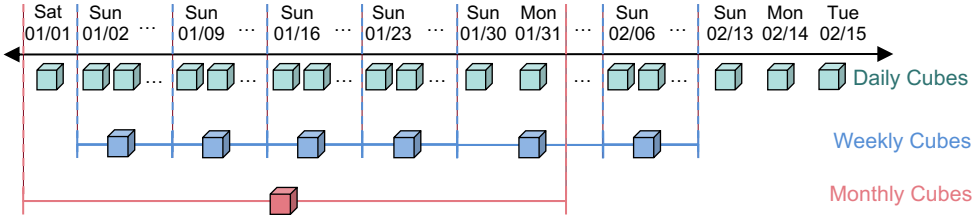


Two Disk Access optimization techniques

Caching & Level Optimization

Example:
Find number of updates in during the period Jan 1 to Feb 15.
(46 days)

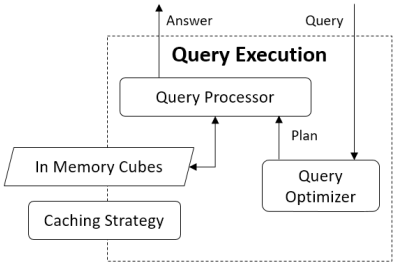
B. Level Optimization



Three Possible Paths:

A- All daily	cost = **46 cubes**
B- 6 weekly + 4 daily	cost = **10 cubes**
C- 1 monthly + 1 weekly + 8 daily	cost = **10 cubes**

3- Query Execution

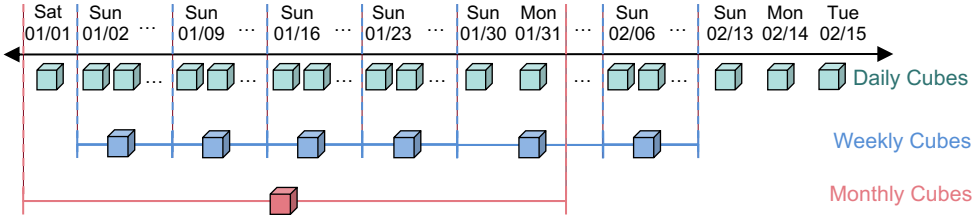


Two Disk Access optimization techniques

Caching & Level Optimization

Example:
Find number of updates in during the period Jan 1 to Feb 15.
(46 days)

B. Level Optimization

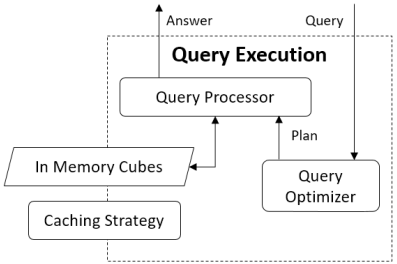


Three Possible Paths:

A- All daily	cost = **46 cubes**
B- 6 weekly + 4 daily	cost = **10 cubes**
C- 1 monthly + 1 weekly + 8 daily	cost = **10 cubes**

Cache Awareness

3- Query Execution

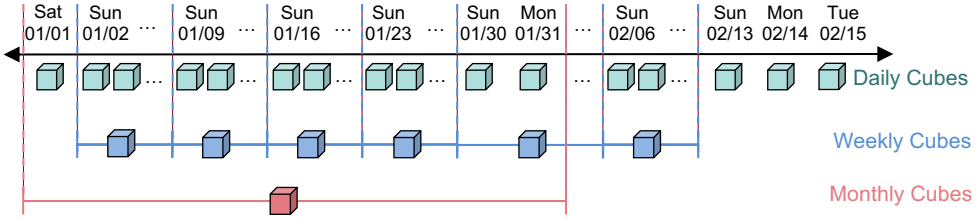


Two Disk Access optimization techniques

Caching & Level Optimization

Example:
Find number of updates in during the period Jan 1 to Feb 15.
(46 days)

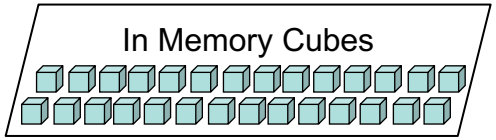
B. Level Optimization



Three Possible Paths:

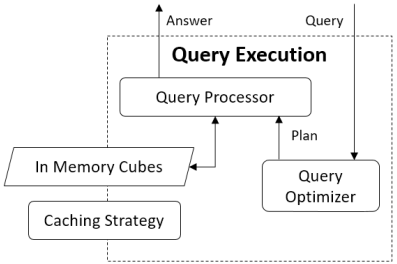
A- All daily	cost = **46 cubes**
B- 6 weekly + 4 daily	cost = **10 cubes**
C- 1 monthly + 1 weekly + 8 daily	cost = **10 cubes**

Cache Awareness



E.g. $\alpha = 1$

3- Query Execution

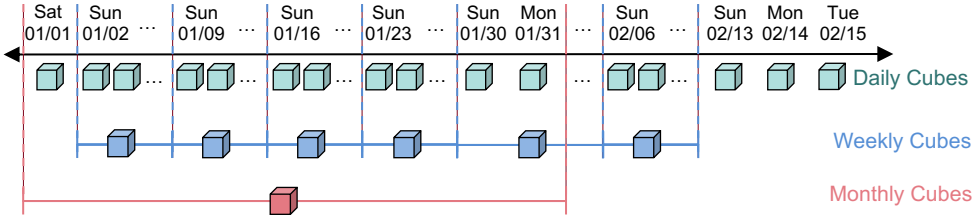


Two Disk Access optimization techniques

Caching & Level Optimization

Example:
Find number of updates in during the period Jan 1 to Feb 15. (46 days)

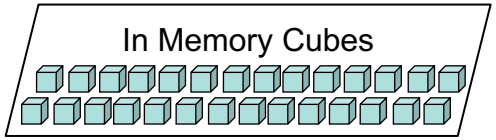
B. Level Optimization



Three Possible Paths:

A- All daily	cost = **46 cubes**
B- 6 weekly + 4 daily	cost = **10 cubes**
C- 1 monthly + 1 weekly + 8 daily	cost = **10 cubes**

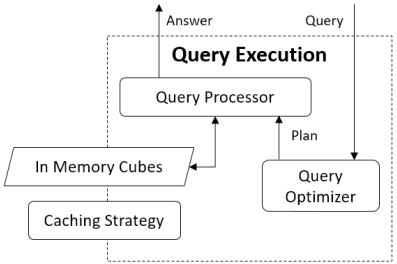
Cache Awareness



E.g. $\alpha = 1$

A- All daily	**cost = 0 disk I/O**
B- 6 weekly + 4 daily	**cost = 6 disk I/O**
C- 1 monthly + 1 weekly + 8 daily	**cost = 2 disk I/O**

3- Query Execution

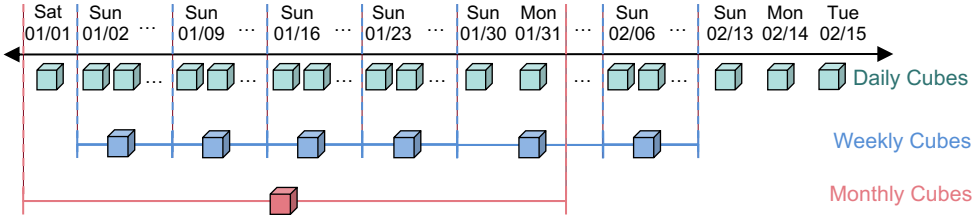


Two Disk Access optimization techniques

Caching & Level Optimization

Example:
Find number of updates in during the period Jan 1 to Feb 15. (46 days)

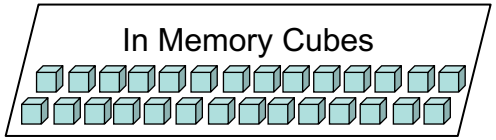
B. Level Optimization



Three Possible Paths:

A- All daily	cost = **46 cubes**
B- 6 weekly + 4 daily	cost = **10 cubes**
C- 1 monthly + 1 weekly + 8 daily	cost = **10 cubes**

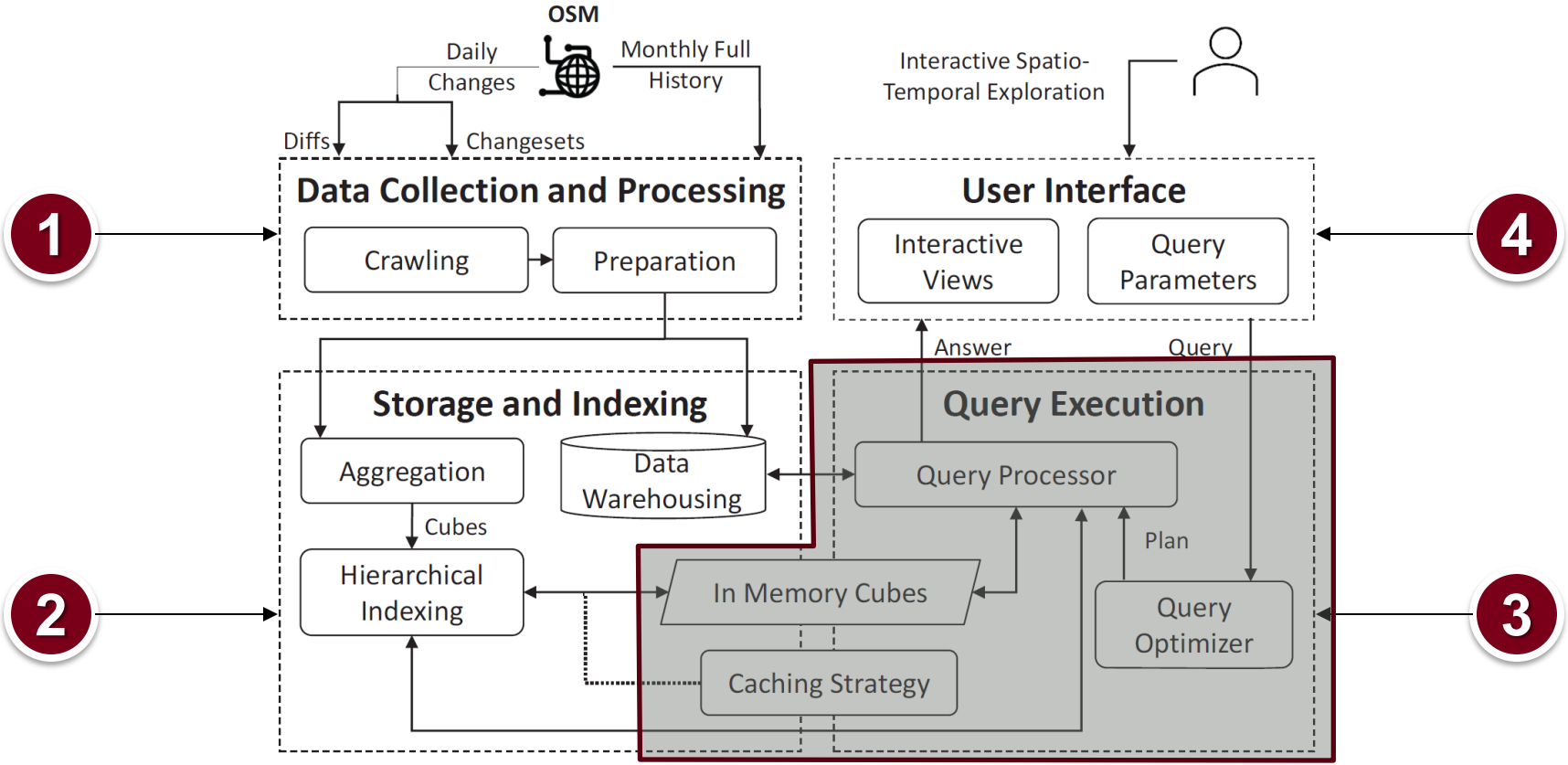
Cache Awareness



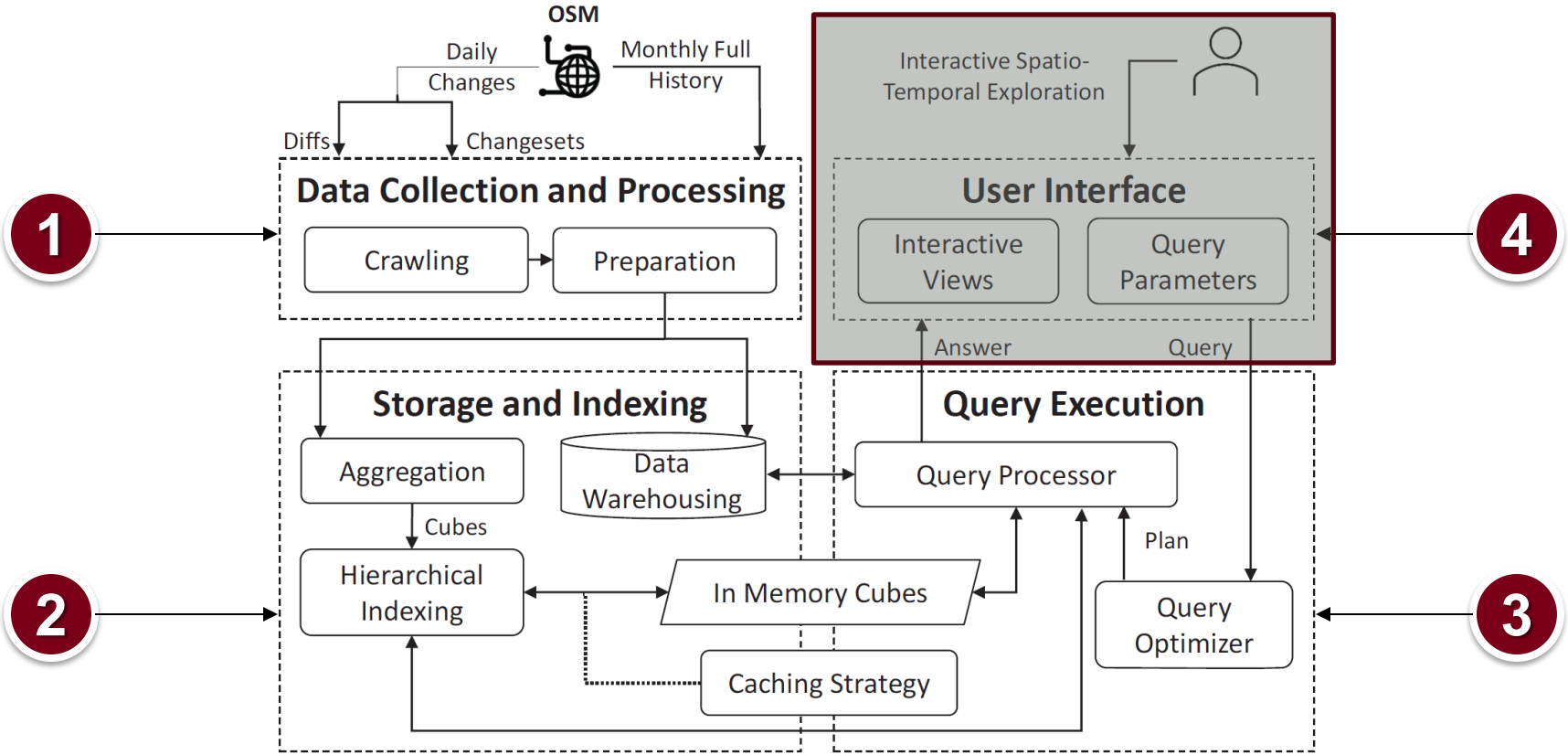
E.g. α = 1

A- All daily	**cost = 0 disk I/O**
B- 6 weekly + 4 daily	**cost = 6 disk I/O**
C- 1 monthly + 1 weekly + 8 daily	**cost = 2 disk I/O**

RASED Architecture



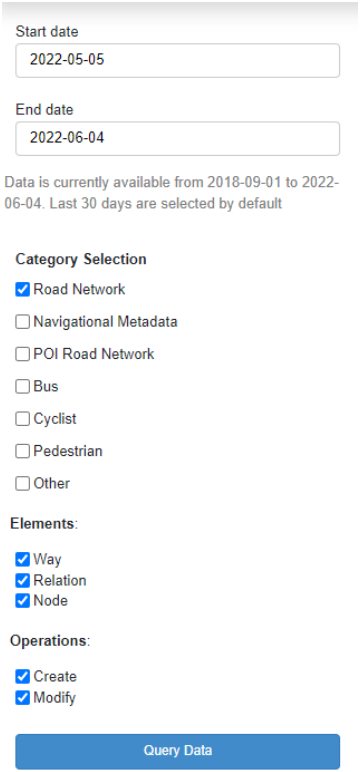
RASED Architecture



3- User Interface

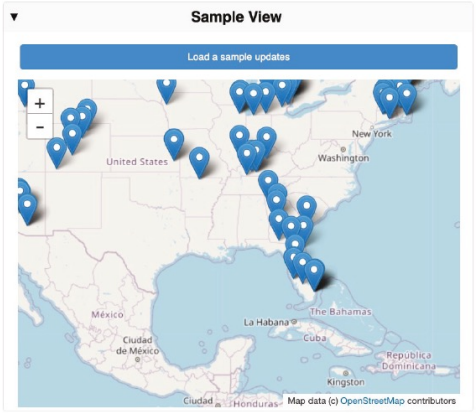
3- User Interface

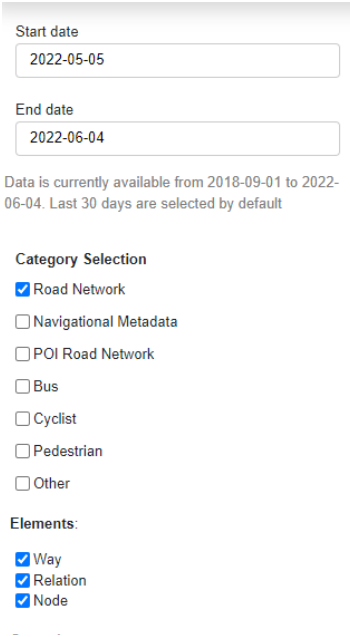
- **Input**: Query Parameters

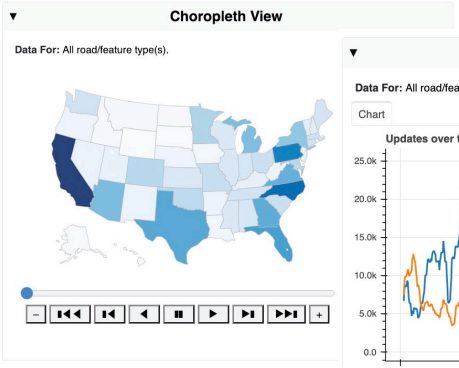


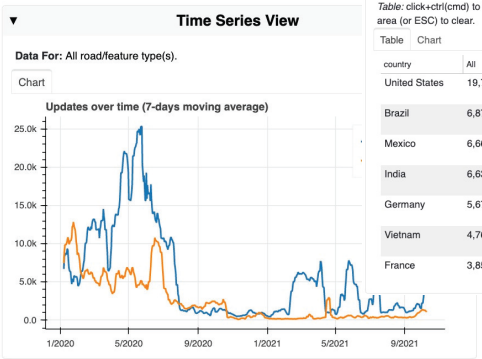
3- User Interface

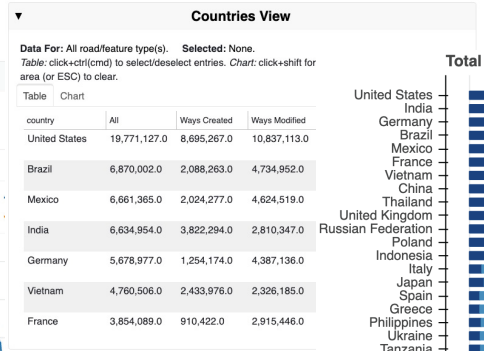
- **Input**: Query Parameters
- **Output**:
  - Statistics either absolute numbers or percentages
  - Visualizations as:
    - Tabular format sorted on any column,
    - Various charts (bar, choropleth, time series)
    - Time-lapse video showing the road network evolution

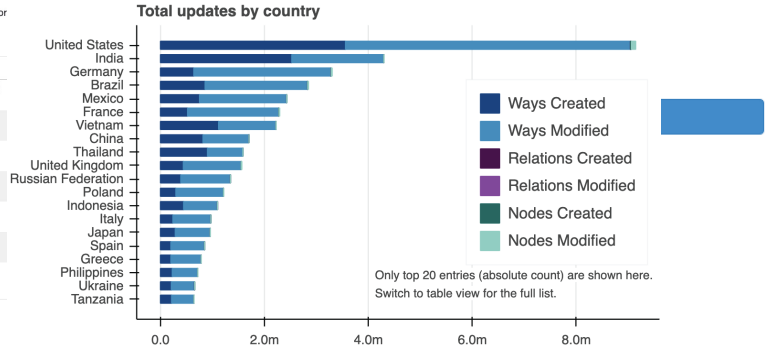










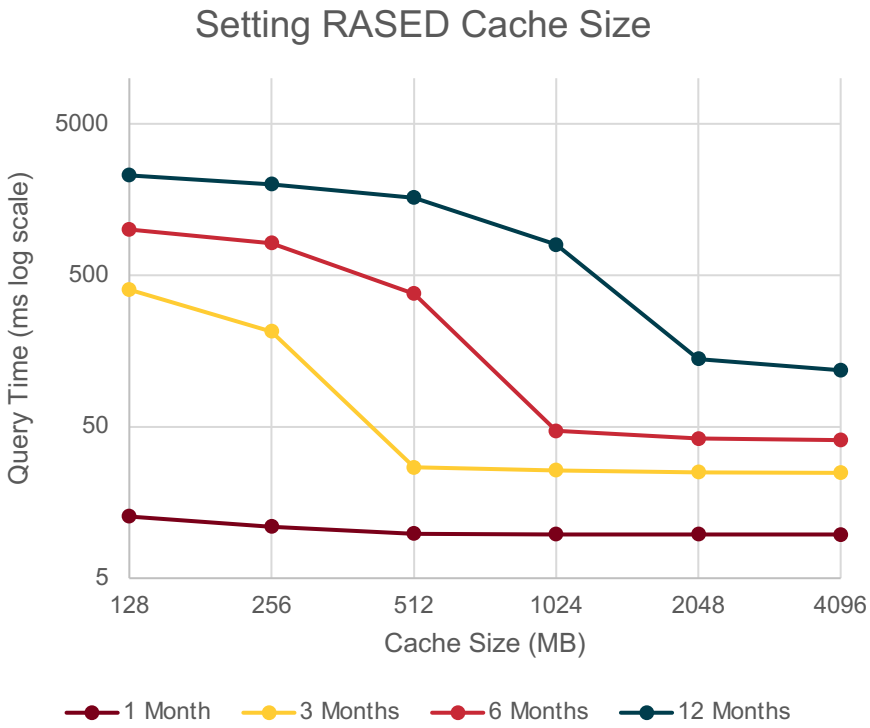


Outline

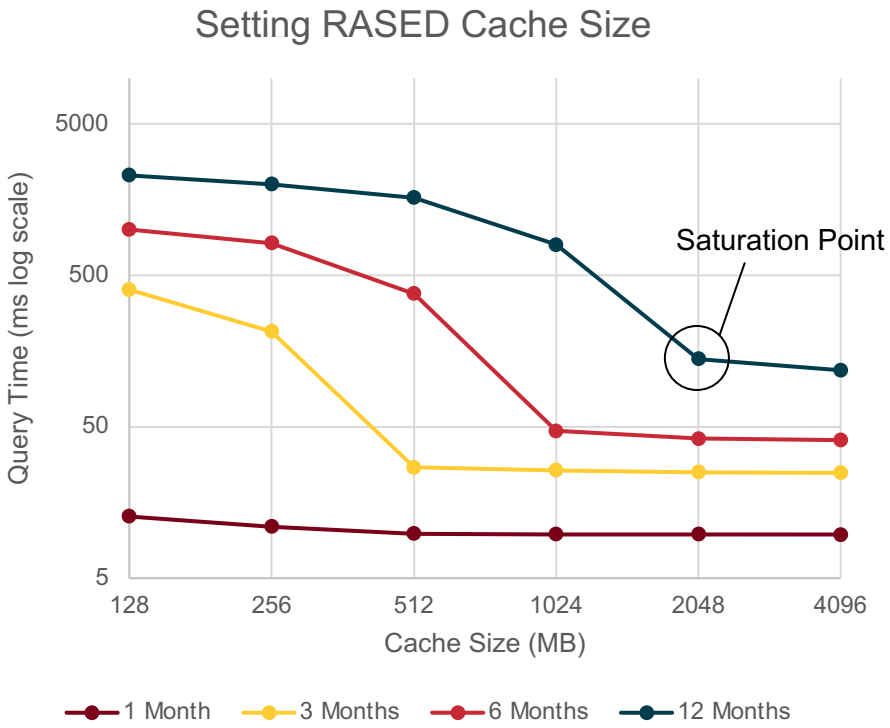
1. **Motivation**
2. **Background**
   - Map updates, Datasets, and Queries
3. **RASED Architecture & Main Modules**
   - Data Collection & Processing
   - Storage & Indexing
   - Query Execution
   - User Interface
4. **Experimental Results**
5. **RASED in Action (quick demo)**

Setting RASED Parameters

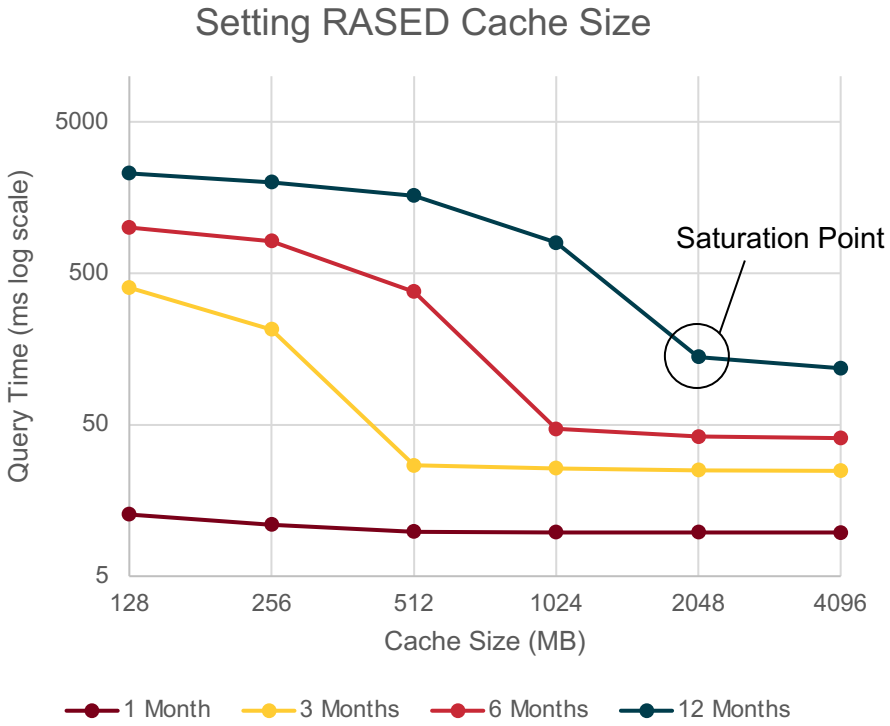
Setting RASED Parameters



Setting RASED Parameters

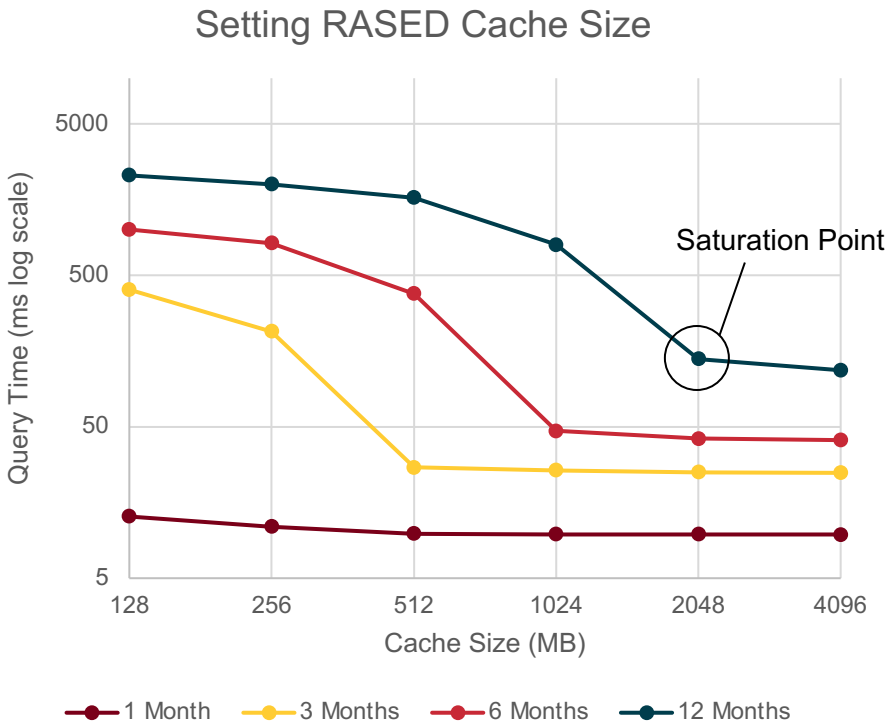


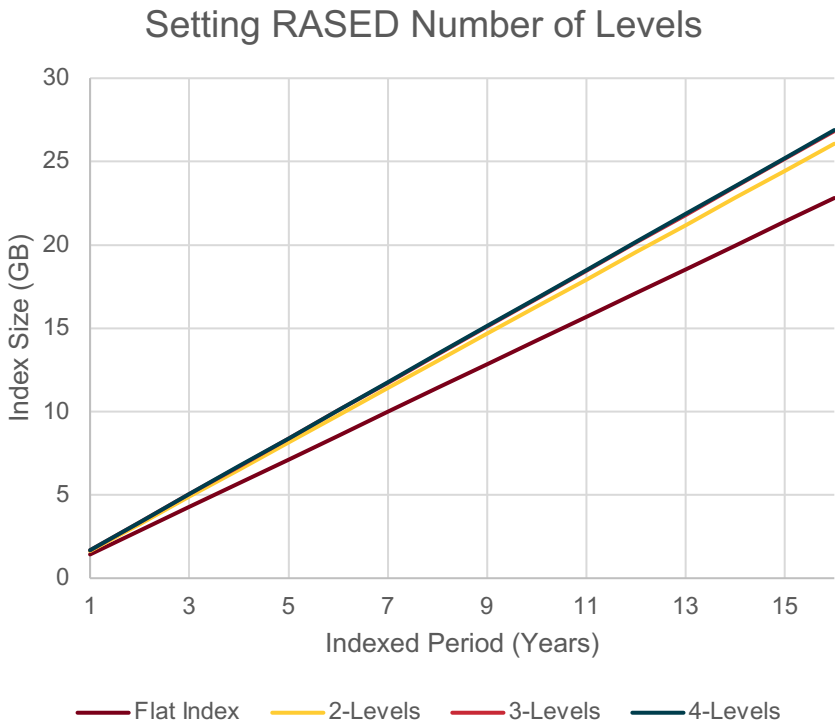
Setting RASED Parameters



RASED Uses 2048 MB for Cache

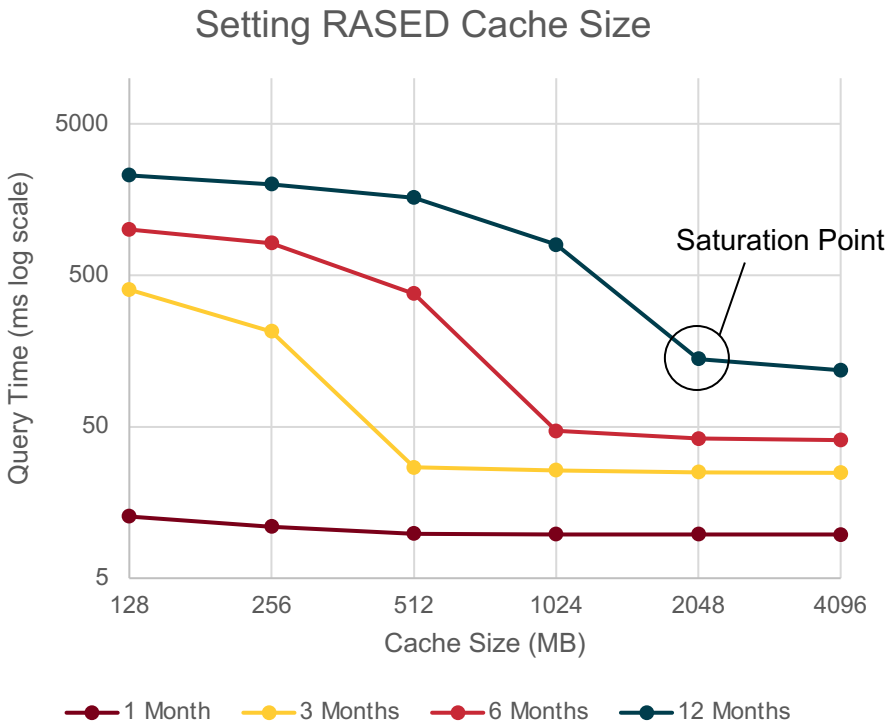
Setting RASED Parameters

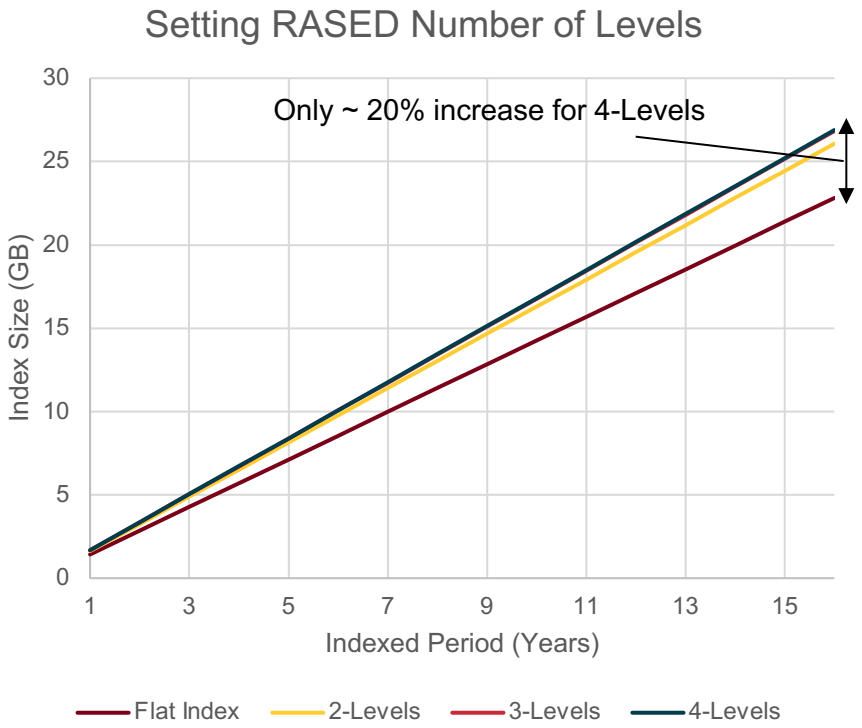




RASED Uses 2048 MB for Cache

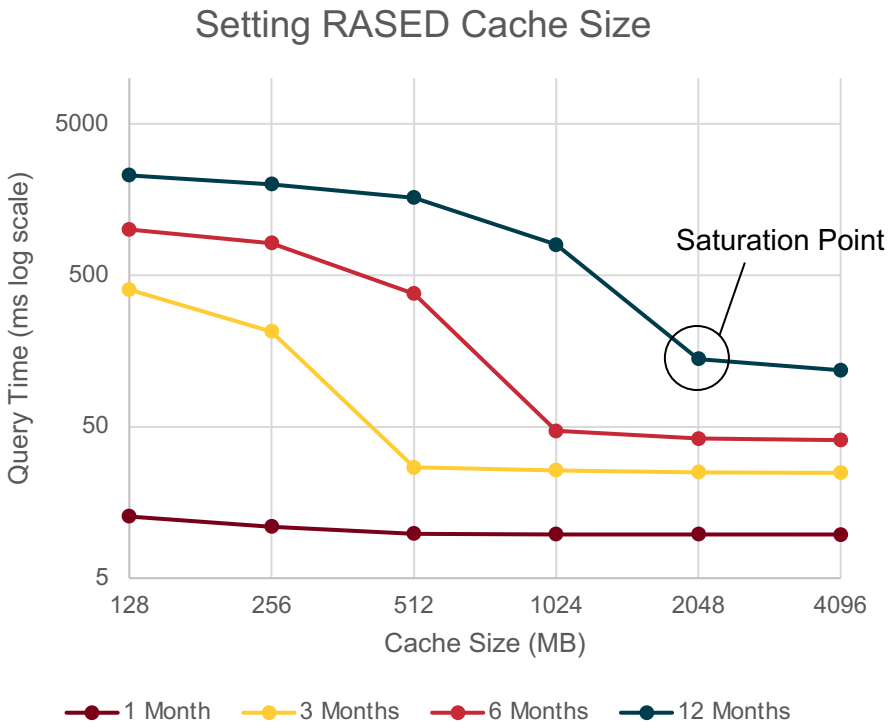
Setting RASED Parameters

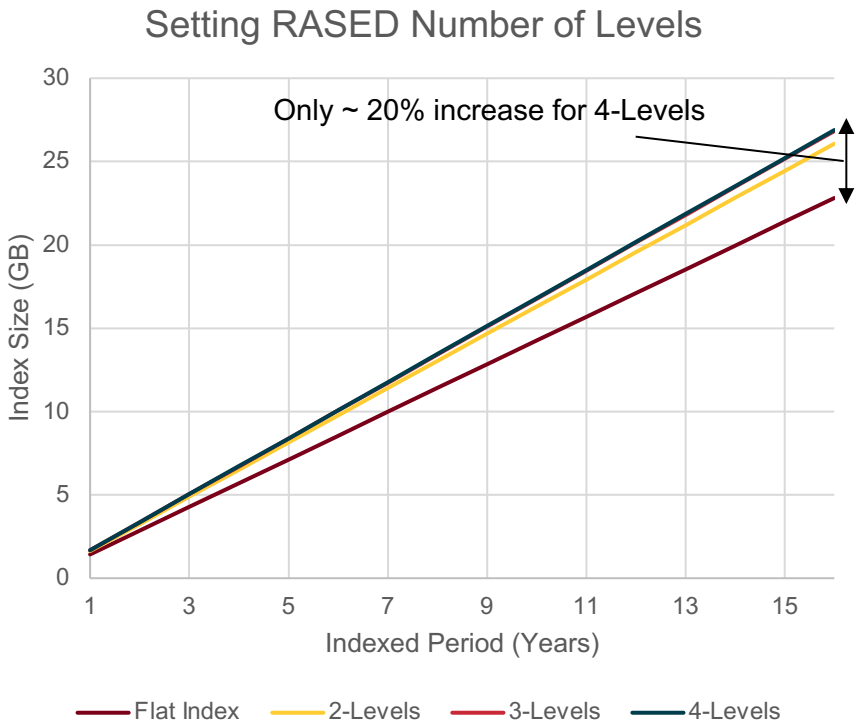




RASED Uses 2048 MB for Cache

Setting RASED Parameters





RASED Uses 2048 MB for Cache

RASED Index Uses 4 Levels

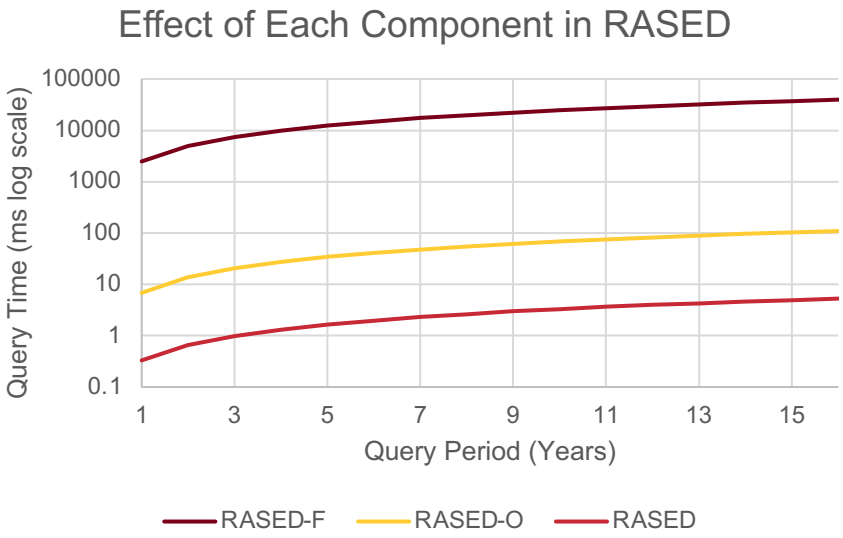
RASED Performance

RASED Performance

RASED-F: one level flat index, neither caching nor level optimization
RASED-O: hierarchal index with level optimization, no caching
RASED: full RASED system with both caching nor level optimization

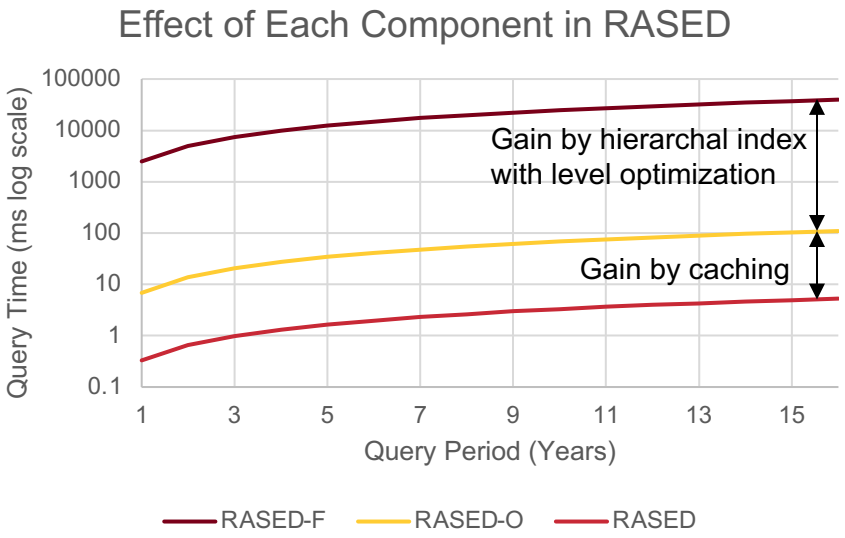
RASED Performance

RASED-F: one level flat index, neither caching nor level optimization
RASED-O: hierarchal index with level optimization, no caching
RASED: full RASED system with both caching nor level optimization



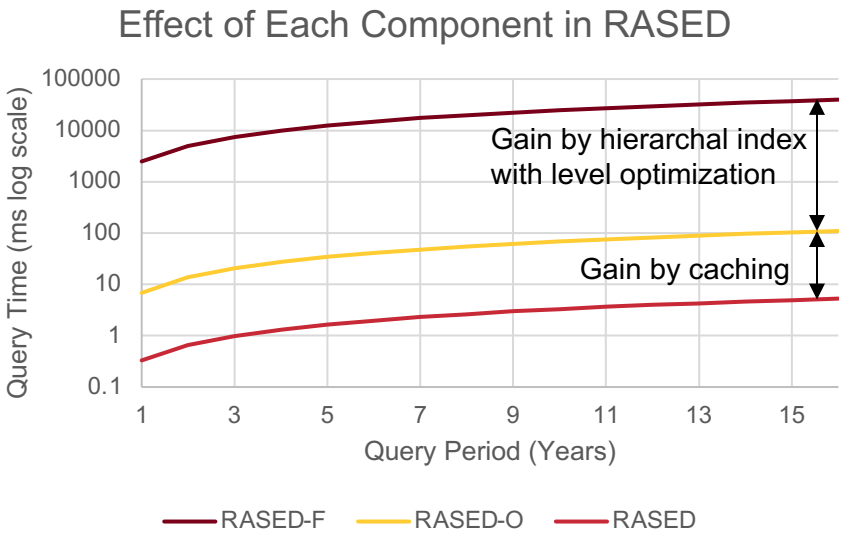
RASED Performance

RASED-F: one level flat index, neither caching nor level optimization
RASED-O: hierarchal index with level optimization, no caching
RASED: full RASED system with both caching nor level optimization



RASED Performance

RASED-F: one level flat index, neither caching nor level optimization
RASED-O: hierarchal index with level optimization, no caching
RASED: full RASED system with both caching nor level optimization

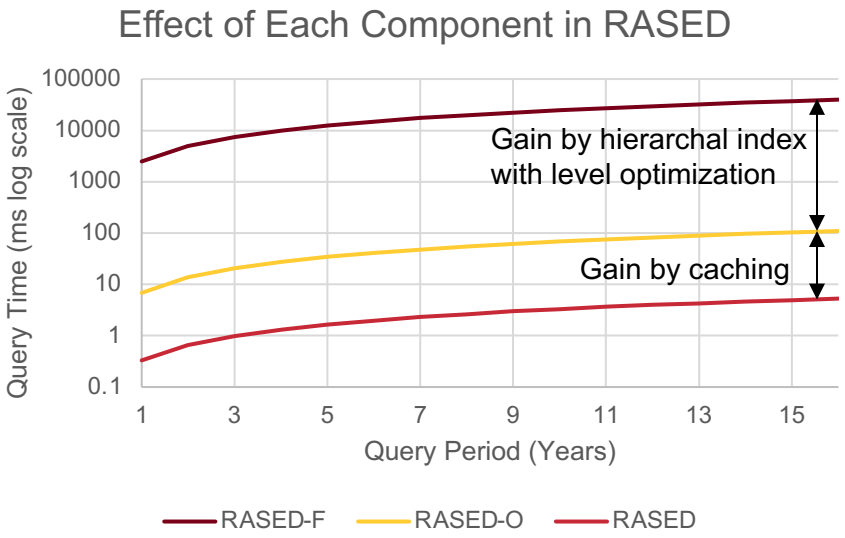


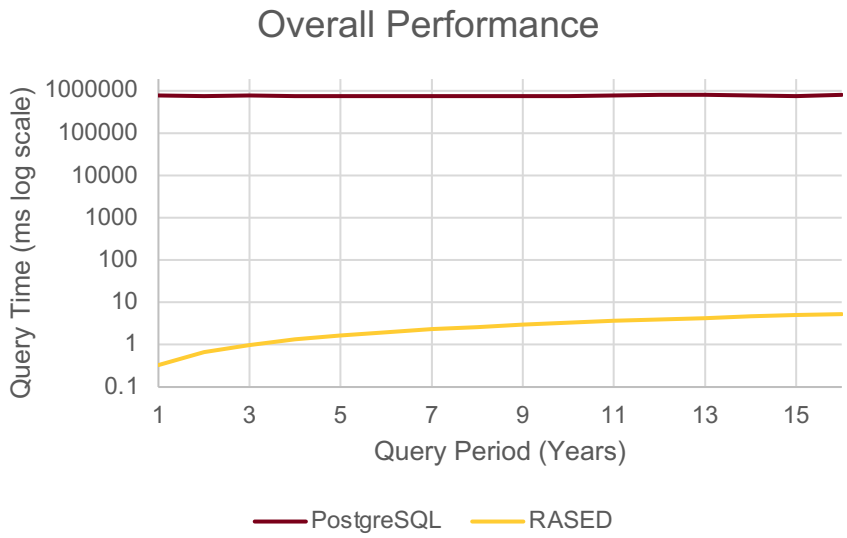
RASED indexing, optimizing, and caching reduce response times by orders of magnitude

RASED Performance

RASED-F: one level flat index, neither caching nor level optimization
RASED-O: hierarchal index with level optimization, no caching
RASED: full RASED system with both caching nor level optimization

PostgreSQL constantly requires scanning the whole data because query involves multiple attributes in the Group By clause





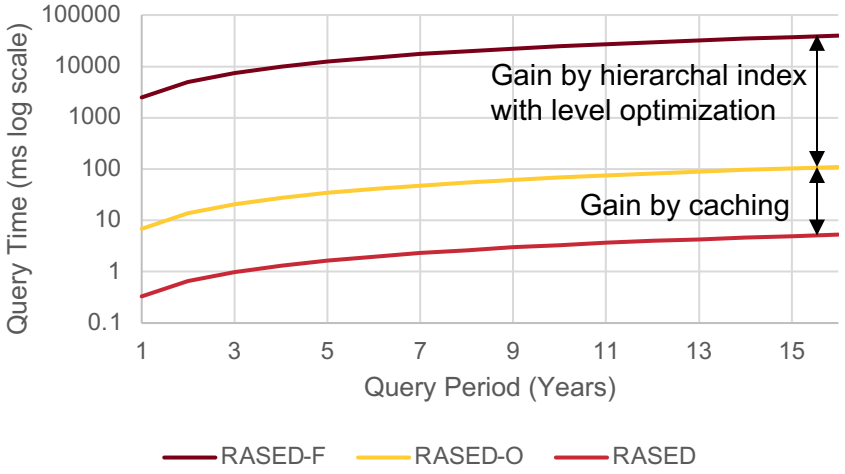
RASED indexing, optimizing, and caching reduce response times by orders of magnitude

RASED Performance

RASED-F: one level flat index, neither caching nor level optimization
RASED-O: hierarchal index with level optimization, no caching
RASED: full RASED system with both caching nor level optimization

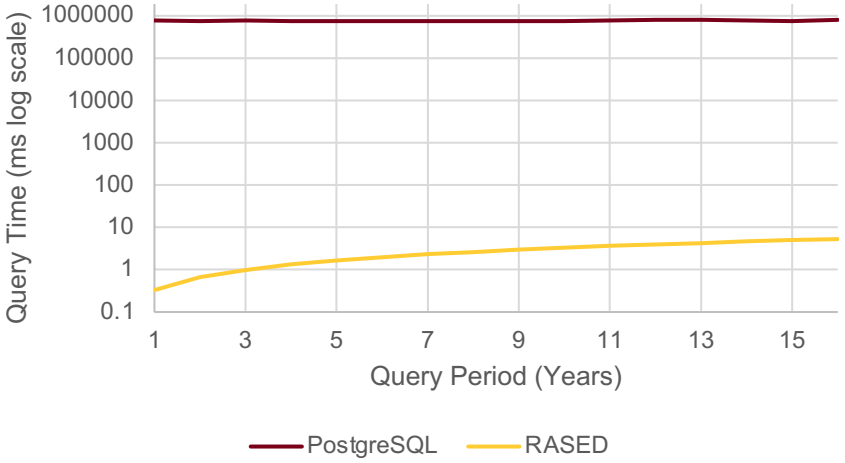
PostgreSQL constantly requires scanning the whole data because query involves multiple attributes in the Group By clause

Effect of Each Component in RASED



RASED indexing, optimizing, and caching reduce response times by orders of magnitude

Overall Performance



RASED queries are always answered in milliseconds

Outline

1. **Motivation**
2. **Background**
   - Map updates, Datasets, and Queries
3. **RASED Architecture & Main Modules**
   - Data Collection & Processing
   - Storage & Indexing
   - Query Execution
   - User Interface
4. **Experimental Results**
5. **RASED in Action (quick demo)**

DEMO

https://rased.cs.umn.edu

https://www.youtube.com/watch?v=fB2JxTZ60wU

THANK YOU