Advanced Pre-aggregations Workshop

cube


Adnan Rahic


Igor Lukanin


Ryan Pei

Community Code of Conduct

- We want to foster an open and welcoming environment where everyone feels they belong in the Cube.js community.
- The full text of our Code of Conduct is available at https://github.com/cube-js/cube.js/blob/master/CODE_OF_CONDUCT.md
- Any instances of inappropriate/unacceptable behavior can be reported to conduct@cube.dev.

Some quick notes 🗒️

- If you have any questions during the workshop, please feel free to type them in the “Q&A”.
- We will be using Cube Cloud for “hands-on” demos.
- Recording of today’s workshop will be posted on the Cube Dev YouTube channel.
- All attendees will receive a post-event survey and we’d appreciate your feedback to help us with future events.
    - Instead of a traditional thank you gift for survey responses, you will have a chance to select a from a list charities that Cube will make a donation to.
        - Doctors Without Borders, UNHCR, UNICEF, and Save the Children

What we will discuss today

- Recap of the first workshop and product updates
- Structuring, optimizing, and partitioning pre-aggregations
- Understanding pre-aggregation cardinality
- 1st Q&A Session
- Advanced strategies for refreshing pre-aggregations
- Rollup Joins and Multi-tenancy best practices
- 2nd Q&A Session

A quick recap of the first workshop

What is a pre-aggregation?

- Condensed versions of source data
- Partitioned, stored, and refreshed — optimized for efficient queries
- Most common type of pre-aggregation is **rollup**
- “Data condensing” mechanism of rollups == **GROUP BY** function in SQL

How do pre-aggregations work?

Raw Data

Order ID	Order Time	Product	Quantity
1	July 20, 2021 10:00 AM	Sugar Cubes	2
2	July 22, 2021 08:00 AM	Ice Cubes	4
3	July 26, 2021 03:00 PM	Ice Cubes	2
4	July 28, 2021 12:00 AM	Sugar Cubes	1
5	August 3, 2021 12:00 PM	Ice Cubes	4
6	August 5, 2021 09:00 AM	Ice Cubes	1

Row Count

~1M (5 years of data)

Pre-Aggregated Data

Order Time	Product	Quantity
July	Ice Cubes	6
July	Sugar Cubes	3
August	Ice Cubes	5

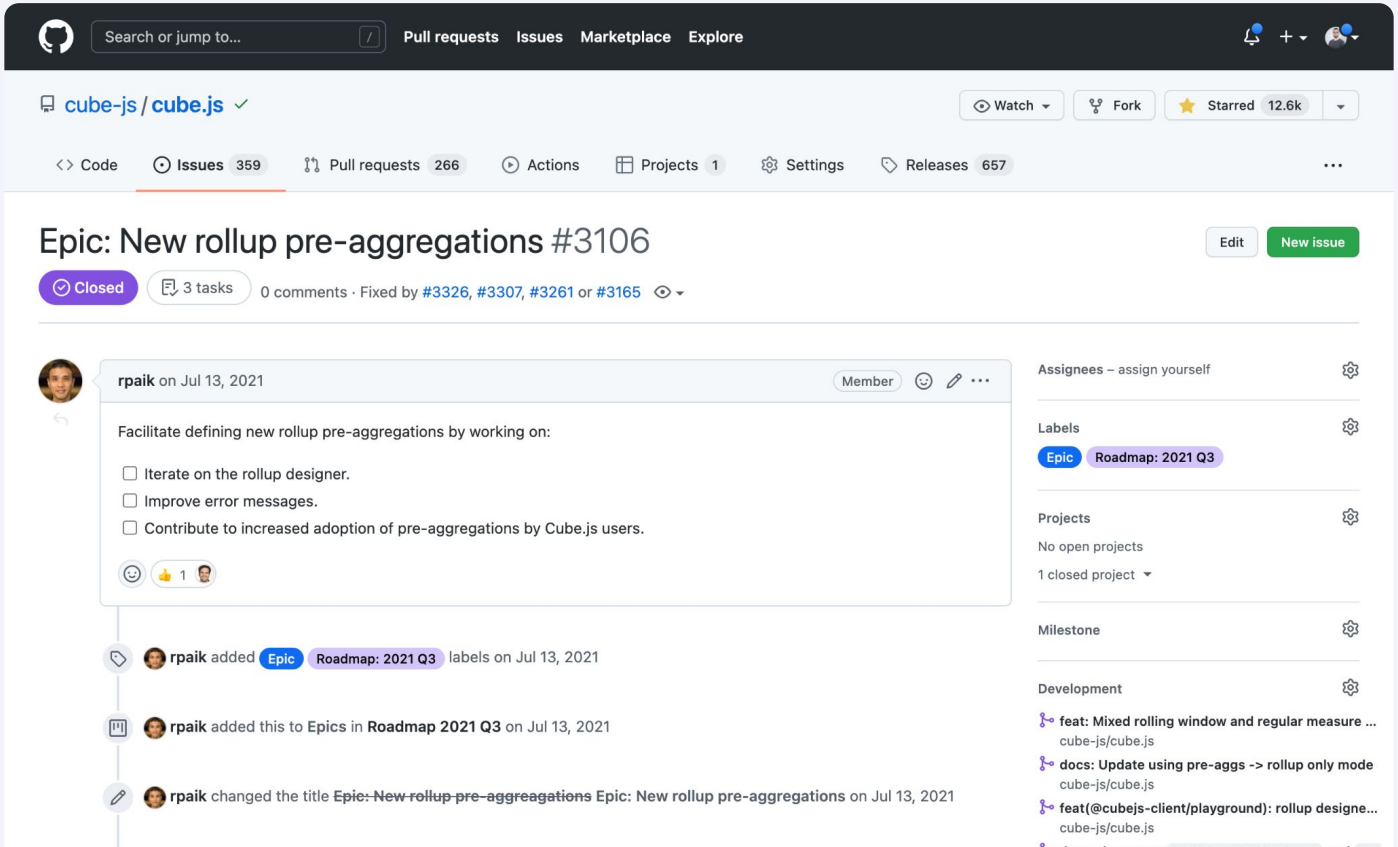
Row Count

~120

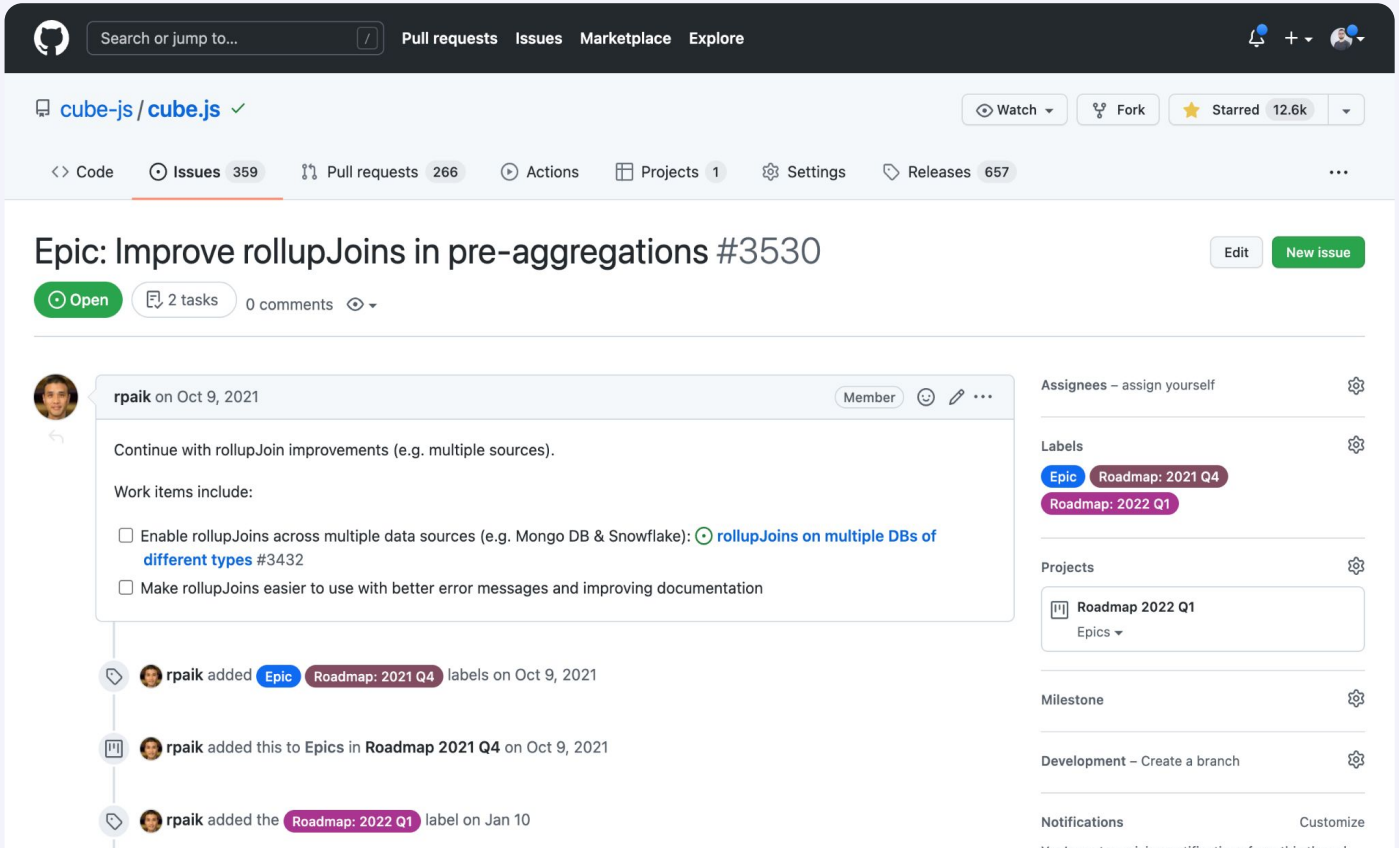
A recap blog post from the first workshop:
https://cube.dev/blog/pre-aggregations-workshop-recap/

Improvements in pre-aggregations since the first workshop

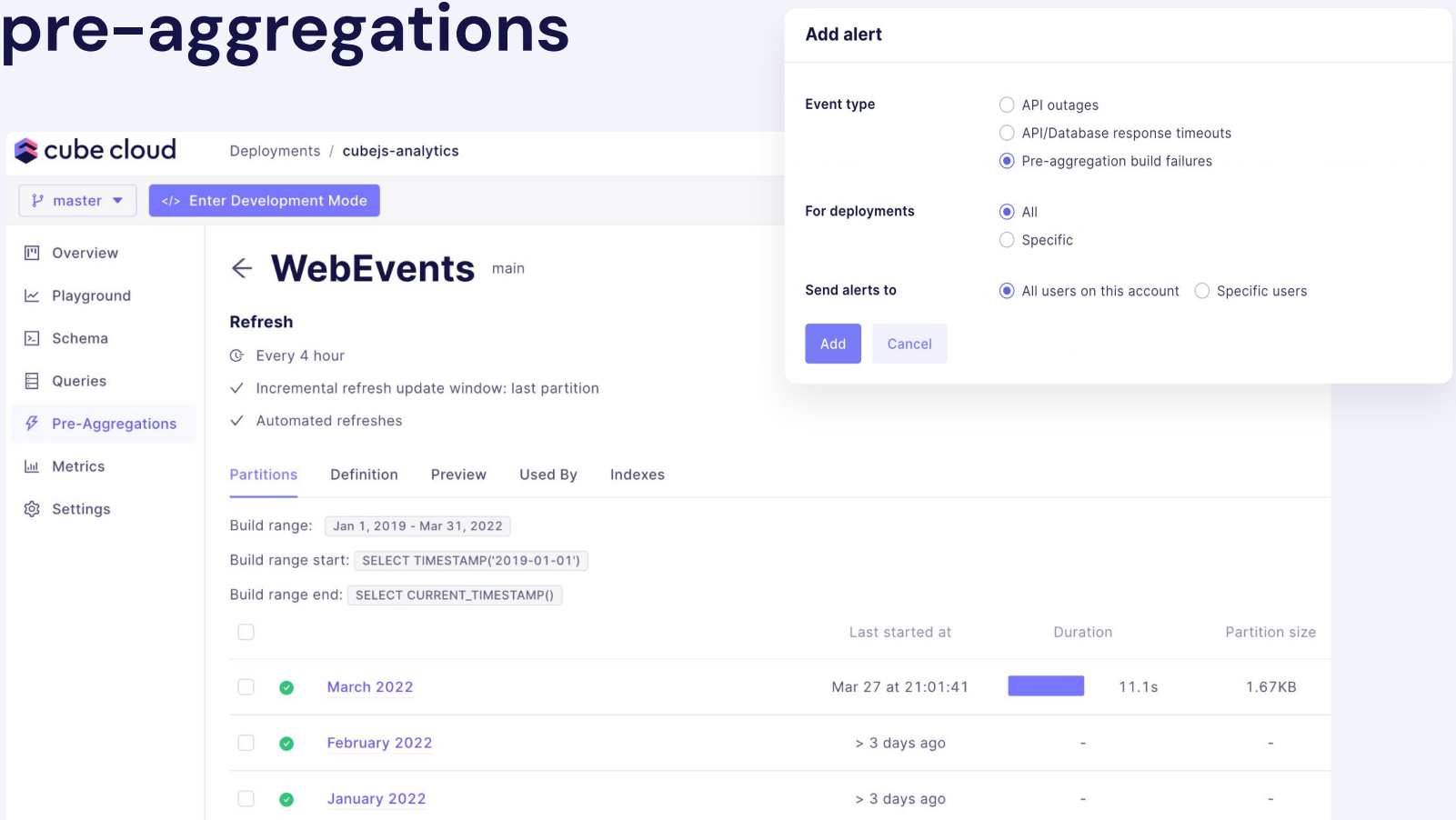
Updates


Search or jump to...
Pull requests
Issues
Marketplace
Explore
cube-js / cube.js
Watch
Fork
Starred 12.6k
Code
Issues 359
Pull requests 266
Actions
Projects 1
Settings
Releases 657
Epic: New rollup pre-aggregations #3106
Edit
New issue
Closed
3 tasks
0 comments · Fixed by #3326, #3307, #3261 or #3165
rpaik on Jul 13, 2021
Member
Facilitate defining new rollup pre-aggregations by working on:
Iterate on the rollup designer.
Improve error messages.
Contribute to increased adoption of pre-aggregations by Cube.js users.
rpaik added Epic Roadmap: 2021 Q3 labels on Jul 13, 2021
rpaik added this to Epics in Roadmap 2021 Q3 on Jul 13, 2021
rpaik changed the title Epic: New rollup pre-aggreagations Epic: New rollup pre-aggregations on Jul 13, 2021
Assignees – assign yourself
Labels
Epic
Roadmap: 2021 Q3
Projects
No open projects
1 closed project
Milestone
Development
feat: Mixed rolling window and regular measure ...
cube-js/cube.js
docs: Update using pre-aggs -> rollup only mode
cube-js/cube.js
feat(@cubejs-client/playground): rollup designe...
cube-js/cube.js

On the roadmap for the coming month



Cube Cloud — more insight into pre-aggregations



Cube Store

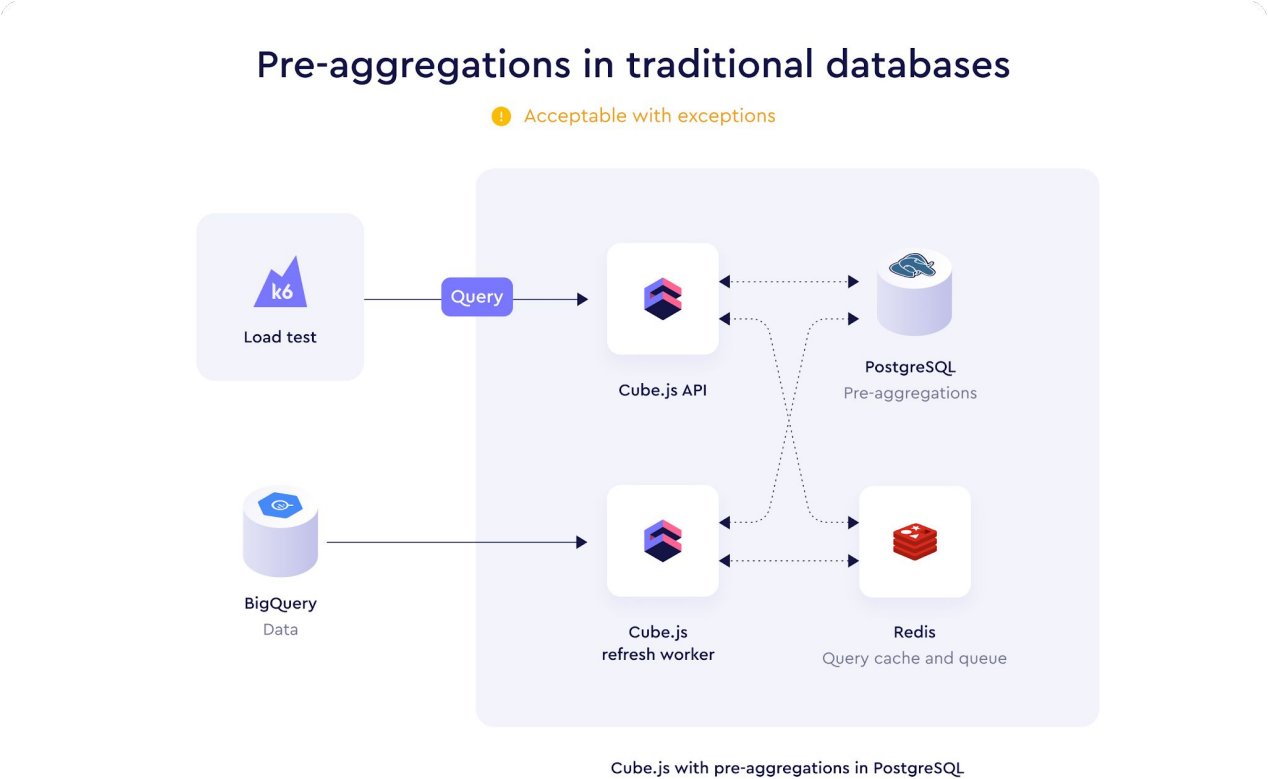
Production checklist

- Cube Store 🚀
- Dedicated Refresh Worker
- Dedicated Redis
- Batching and/or exporting data
- Pre-aggregation build/refreshes may need tuning

Before Cube Store

- Pre-aggregations stored in traditional databases
- **Pre-aggregations in traditional databases often won't allow for high concurrency and low latency of the analytical API**

Before Cube Store

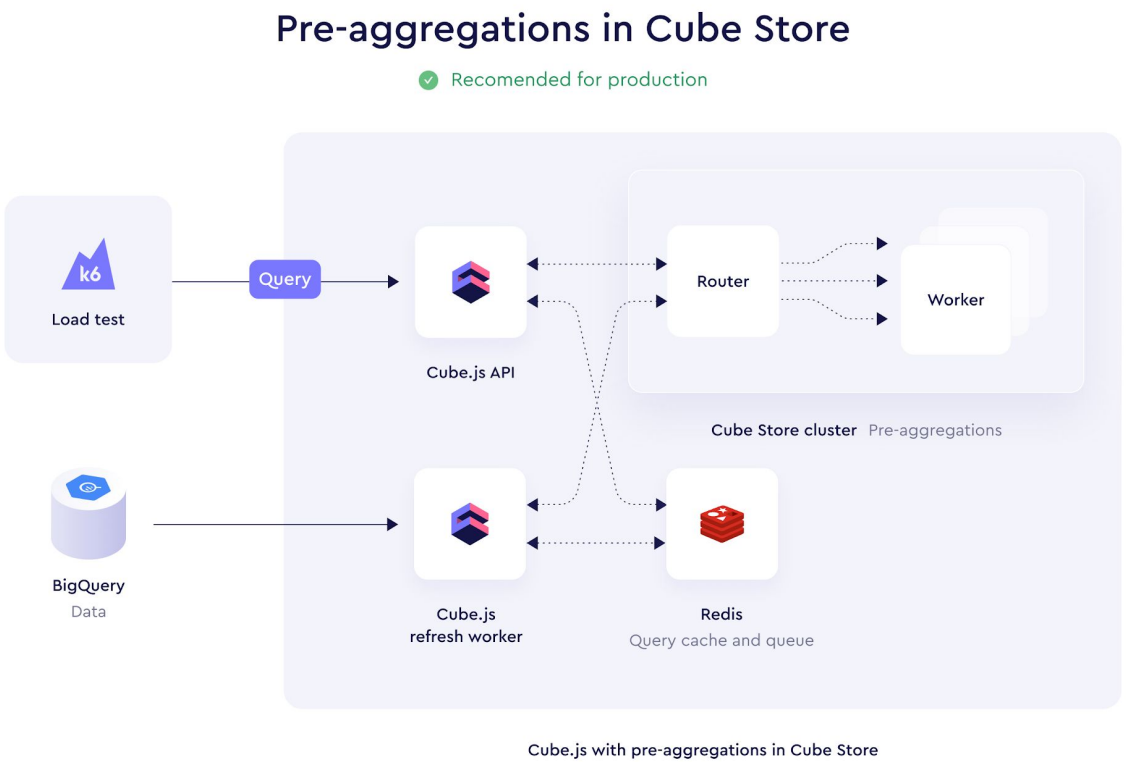


Cube.js with pre-aggregations in PostgreSQL

After Cube Store

- Cube Store is designed to resolve these issues and provide a performant pre-aggregation storage layer for Cube

After Cube Store

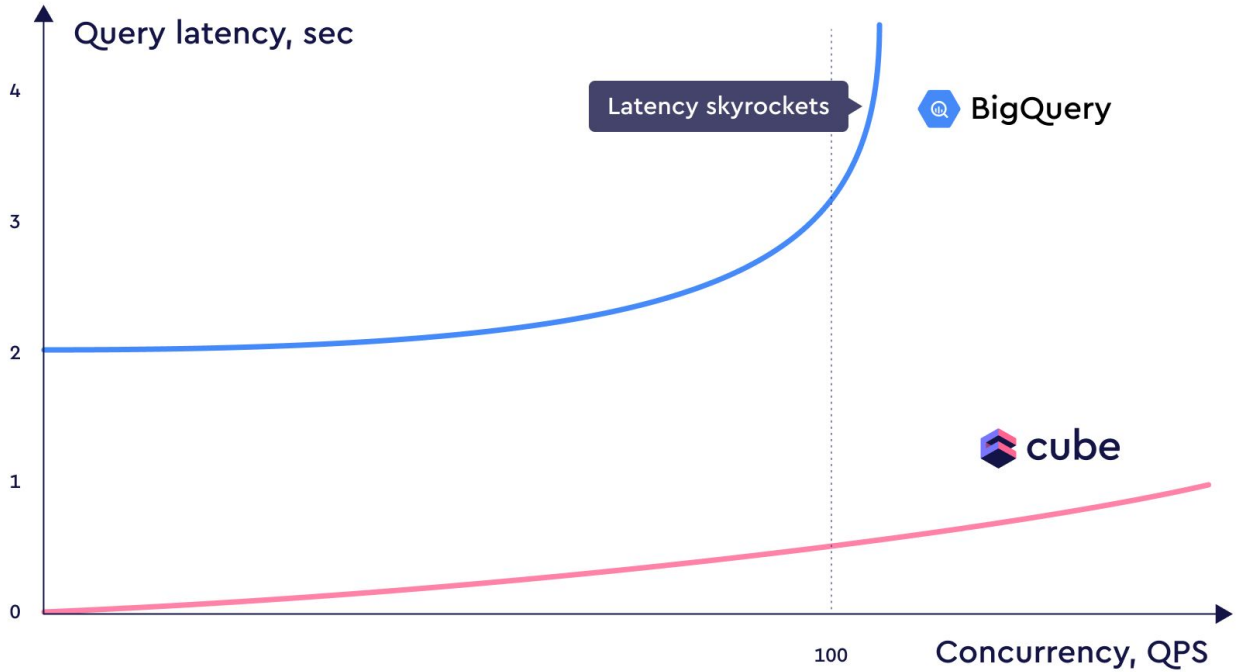


Cube.js with pre-aggregations in Cube Store

What and why Cube Store?

- Significant performance improvement
- Significantly lower latency
- Latency decreased by 5x in this use-case:
  - https://cube.dev/blog/data-warehouse-performance-and-how-cube-can-help/

Why Cube Store?



TPC-H BigQuery dataset with >150M entries

Prep Guide

- https://descriptive-reply-0b7.notion.site/Workshop-Advanced-Pre-aggregations-Prep-Guide-5de1ae39a28849edb68d29f29728bc7c

Code

- https://descriptive-reply-0b7.notion.site/Workshop-Advanced-Pre-aggregations-Code-b08909ee5ecd43a1b7de4baa70f8cf28

Download the GCP service account key

Use this key to access BigQuery

- https://buff.ly/3qJFI16

TPC-H x 100 dataset

- >150M entries
- BigQuery schema called `tpc_h` in the `cube-devrel-team` BigQuery instance

Demo: Step 0
BigQuery and TPC-H dataset

Matching Additive vs Non-Additive measures with Pre-aggregations

Additive vs Non-Additive measures

- Non-additive measures are average values or distinct counts
- Pre-aggregations with these measures are less likely to be selected to accelerate a query

Non-additive measures

```
totalPriceAvg: {
  sql: `${CUBE}.O_TOTALPRICE`,
  type: `avg`
},

clerkCountDistinct: {
  sql: `${CUBE}.O_CLERK`,
  type: `countDistinct`
},
```

Matching non-additive measures with pre-aggregations

```
totalPriceAvgClerkCountDist: {
  measures: [
    Order.totalPriceAvg,
    Order.clerkCountDistinct
  ],
  dimensions: [
    // Will NOT match without this dimension
    Order.oOrderpriority
  ]
},
```

Turning non-additive into additive measures

```
totalPriceAvg: {
  sql: `${CUBE.totalPriceSum} / ${CUBE.count}`,
  type: `number`,
},
totalPriceSum: {
  sql: `${CUBE}.O_TOTALPRICE`,
  type: `sum`,
},
clerkCountDistinct: {
  sql: `${CUBE}.O_CLERK`,
  type: `countDistinctApprox`
},
```

Matching additive measures

```
totalPriceAvgClerkCountDist: {
  measures: [
    Order.totalPriceSum,
    Order.count,
    Order.clerkCountDistinct
  ],
  dimensions: [
    // Will match even without this dimension
    Order.oOrderpriority
  ]
}
```

Demo: Step 1
Matching non-additive and additive measures with pre-aggregations

Providing multiple pre-aggregations to solve issues with additive vs non-additive measures

Providing multiple pre-aggregations

- Dedicated pre-aggregations
- Definitions that fully match queries with non-additive measures
- Performance boost but longer build time and space consumed

Providing multiple pre-aggregations

```
/** With a dimension */
totalPriceAvgClerkCountDist1: {
  measures: [
    Order.totalPriceAvg,
    Order.clerkCountDistinct
  ],
  dimensions: [
    Order.oOrderpriority
  ]
},
/** Without a dimension */
totalPriceAvgClerkCountDist2: {
  measures: [
    Order.totalPriceAvg,
    Order.clerkCountDistinct
  ]
},
```

Demo: Step 2
Providing multiple pre-aggregations to solve issues with additive vs non-additive measures

Pre-aggregations with large queries that have joins

Pre-aggregations with large queries and joins

- Pre-aggregations across joined cubes
- Produces large result sets
- Produces large pre-aggregations

Let’s run a large query

```
{
  "measures": [
    "Order.count",
    "Order.totalPriceAvg",
    "Order.totalPriceSum"
  ],
  "timeDimensions": [
    {
      "dimension": "Order.oOrderdate",
      "granularity": "day",
      "dateRange": [
        "1998-08-01",
        "1998-08-02"
      ]
    }
  ],
  "order": {
    "Order.count": "desc"
  },
  "dimensions": [
    "Customer.cName"
  ],
  "limit": 10
}
```

Pre-aggregation definition

```
dailyOrdersPerCustomer: {
  measures: [
    Order.count,
    Order.totalPriceSum
  ],
  dimensions: [
    Customer.cName
  ],
  timeDimension: Order.oOrderdate,
  granularity: `day`
}
```

Demo: Step 3
Pre-aggregations with large queries that have joins

Demo: Step 3

- Add a join and a few dimensions first
- Add pre-aggregations
- Once the pre-aggregations start building we will see how long it will take due to the large dataset and no implemented partitions and optimization
- You'll see the warning:
  ```
  The pre-aggregation "Order.dailyOrdersPerCustomer" has more than 100000 rows. Please consider using an export bucket. Learn more.
  ```

How to structure and optimize pre-aggregations

Partitioning

- `partitionGranularity` — Cube will generate separate tables for each partition of data
  - `day`, `month`, etc...
- In the demo we'll use:

```
partitionGranularity: `day`
```

What about incremental refresh?

- Incrementally refresh partitioned — `incremental: true`
- Defaults to `false`
- Building partitioned tables separately – slower than building one table
- Set `incremental: true` to refresh the last partition only
- Set `updateWindow: \`7 day\`` to refresh partitions where the end date lies within the `updateWindow` from the current time

Only refresh partitions within the last 7 days

```
refreshKey: {
  every: `1 hour`,
  incremental: true,
  updateWindow: `7 day`,
},
```

Pre-aggregation definition

```
dailyOrdersPerCustomer: {
  measures: [
    Order.count,
    Order.totalPriceSum
  ],
  dimensions: [
    Customer.cName
  ],
  timeDimension: Order.oOrderdate,
  granularity: `day`,
  partitionGranularity: `day`,
  refreshKey: {
    every: `1 hour`,
    incremental: true,
    updateWindow: `7 day`,
  }
}
```

Build Strategies

- Simple
- Batching
- Export Bucket

Enable export bucket

```
CUBEJS_DB_EXPORT_BUCKET=cube_devrel_team_tpch
CUBEJS_DB_EXPORT_BUCKET_TYPE=gcp
```

Demo: Step 4
How to structure and optimize pre-aggregations

Demo: Step 4

- Add the export bucket env vars
- Add a pre-aggregation with `partitionGranularity`
- Add incremental updates with `incremental: true`
- Build a few partitions to see how quick the builds are

Dedicated pre-aggregations for large queries

Dedicated pre-aggregations for large queries

- Multiple queries can match one larger pre-aggregation
- Split pre-aggregations based on measures and dimensions
- Make sure queries exactly match pre-aggregations

Dedicated pre-aggregations for large queries

- Individual size is smaller and the individual build time is shorter with two dedicated pre-aggregations
- Response times are faster with two dedicated pre-aggregations
- Single larger pre-aggregation is in total smaller than the two smaller combined
- Response times are slower with a single larger pre-aggregation

Let’s run a large query

```
{
  "measures": [
    "Order.count",
    "Order.totalPriceAvg",
    "Order.totalPriceSum"
  ],
  "timeDimensions": [
    {
      "dimension": "Order.oOrderdate",
      "granularity": "day",
      "dateRange": [
        "1998-08-01",
        "1998-08-02"
      ]
    }
  ],
  "order": {
    "Order.count": "desc"
  },
  "dimensions": [
    "Customer.cName",
    "Customer.cAcctbal"
  ],
  "limit": 10
}
```

Pre-aggregation definition

```
dailyOrdersPerCustomer: {
  measures: [
    Order.count,
    Order.totalPriceSum
  ],
  dimensions: [
    Customer.cName,
    Customer.cAcctbal
  ],
  timeDimension: Order.oOrderdate,
  granularity: `day`,
  partitionGranularity: `day`,
  refreshKey: {
    every: `1 hour`,
    incremental: true,
    updateWindow: `7 day`,
  }
},
```

Let’s run a smaller query

```
{
  "measures": [
    "Order.count"
  ],
  "timeDimensions": [
    {
      "dimension": "Order.oOrderdate",
      "granularity": "day",
      "dateRange": [
        "1998-08-01",
        "1998-08-02"
      ]
    }
  ],
  "order": {
    "Order.count": "desc"
  },
  "dimensions": [
    "Customer.cName"
  ],
  "limit": 10
}
```

Create multiple dedicated pre-aggregations

```
dailyOrderCountPerCustomer: {
  measures: [
    Order.count,
  ],
  dimensions: [
    Customer.cName
  ],
  timeDimension: Order.oOrderdate,
  granularity: `day`,
  partitionGranularity: `day`,
  refreshKey: {
    every: `1 hour`,
    incremental: true,
    updateWindow: `7 day`,
  }
},
```

```
dailyOrderPriceAvgPerCustomer: {
  measures: [
    Order.totalPriceAvg
  ],
  dimensions: [
    Customer.cAcctbal
  ],
  timeDimension: Order.oOrderdate,
  granularity: `day`,
  partitionGranularity: `day`,
  refreshKey: {
    every: `1 hour`,
    incremental: true,
    updateWindow: `7 day`,
  }
},
```

Demo: Step 5
Dedicated pre-aggregations for large queries

Demo: Step 5

- Add a compound pre-aggregation
- View build time, partition size, and response time
- Split it into two dedicated pre-aggregations
- View improved build time, smaller partition size, and quicker response time

Refresh tuning

Refreshing Pre-aggregations

Trigger refresh with `refreshKey` including:

- `sql`
- `every`
- both `sql` and `every`

Refresh options

- Interval with `every`
- Default value is 1 hour

```
refreshKey: {
  every: `1 hour`, // This will refresh every hour
  incremental: true,
  updateWindow: `7 day`,
},
```

Refresh options

- CRON syntax with `every`
- Default value is 1 hour

```
refreshKey: {
  every: `0 * * * *`, // This will refresh every hour
  incremental: true,
  updateWindow: `7 day`,
},
```

Refresh options

- Custom refresh check SQL with `sql` parameter
- Default `every` value is `2 minute` for BigQuery
- `MAX(updated_at_timestamp)` is generally best practice
- `FILTER_PARAMS` to filter values during SQL generation
- Mimic the incremental update with a custom `WHERE` clause

Refresh options

```
refreshKey: {
  sql: `
  SELECT
    MAX(O_UPDATEDAT) FROM tpc_h.order
  WHERE ${FILTER_PARAMS.Cube.createdAt.filter('O_CREATED_AT')}
  `,
  every: `1 hour`,
},
```

Demo: Step 6
Refresh tuning

Demo: Step 6

- Show 3 approaches to using the `refreshKey` parameter

Rollup-joins

Rollup–joins

- Joins between separate pre-aggregations
- Use pre-aggregations instead of running SQL queries

Pre-aggregation with joined cubes

```
dailyOrderCountPerCustomer: {
  measures: [
    Order.count,
  ],
  dimensions: [
    Customer.cName
  ],
  timeDimension: Order.oOrderdate,
  granularity: `day`,
  partitionGranularity: `day`,
  refreshKey: {
    every: `1 hour`,
    incremental: true,
    updateWindow: `7 day`,
  }
},
```

Rollup-join pre-aggregation definition

- First the orders rollup

```
ordersRollup: {
  measures: [
    Order.count
  ],
  dimensions: [
    Order.oCustkey
  ],
  timeDimension: Order.oOrderdate,
  granularity: `day`,
  partitionGranularity: `day`,
  refreshKey: {
    every: `1 hour`,
    incremental: true,
    updateWindow: `7 day`,
  }
},
```

Rollup-join pre-aggregation definition

- Next the customers

```
customersRollup: {
  dimensions: [
    Customer.cCustkey,
    Customer.cName,
  ],
},
```

Rollup-join pre-aggregation definition

- Join the rollups

```
ordersCustomersRollupJoin: {
  type: `rollupJoin`,
  measures: [Order.count],
  dimensions: [Customer.cName],
  timeDimension: Order.oOrderdate,
  granularity: `day`,
  partitionGranularity: `day`,
  refreshKey: {
    every: `1 hour`,
    incremental: true,
    updateWindow: `7 day`,
  },
  rollups: [
    Customer.customersRollup,
    Order.ordersRollup,
  ],
},
```

Run the same query — same response!

```
{
  "measures": [
    "Order.count"
  ],
  "timeDimensions": [
    {
      "dimension": "Order.oOrderdate",
      "granularity": "day",
      "dateRange": [
        "1998-08-01",
        "1998-08-02"
      ]
    }
  ],
  "order": {
    "Order.count": "desc"
  },
  "dimensions": [
    "Customer.cName"
  ],
  "limit": 10
}
```

Demo: Step 7
Rollup-joins

Demo: Step 7

- Transform a pre-aggregation with joined cubes into a rollup-join
- View the built pre-aggregations in the Cube Cloud UI
- Show how the queries are not hitting the database,
  instead only hitting Cube Store

Multi-tenancy with pre-aggregations

Enforcing tenant-aware filters on all queries

- `queryRewrite`
- Filter queries per tenant
- Tenant will be `region`

Run a query to count orders per customer for a particular region

```
{
  "measures": [
    "Order.count"
  ],
  "timeDimensions": [
    {
      "dimension": "Order.oOrderdate",
      "granularity": "month",
      "dateRange": [
        "1996-01-01",
        "1996-12-31"
      ]
    }
  ],
  "order": {
    "Order.count": "desc"
  },
  "dimensions": [
    "Customer.cName",
    "Region.rName"
  ],
  "limit": 10
}
```

Filter all queries based on the securityContext

```
queryRewrite: (query, { securityContext }) => {
  // Ensure that the security context has the `rRegionkey` property
  if (!securityContext.rRegionkey) {
    throw new Error('No Region Key found in Security Context!');
  }

  // Apply a filter to all queries.  Cube will make sure to join
  // the `Region` cube to other cubes in a query to apply the filter
  query.filters.push({
    member: "Region.rRegionkey",
    operator: "equals",
    values: [ securityContext.rRegionkey ]
  });

  return query;
},
```

Demo: Step 9 / Part 1
Enforcing tenant-aware filters on all queries

Demo: Step 9 / Part 1

- Filter queries based on the `securityContext`
- Add required joins
- Add `queryRewrite` in the `cube.js` file

Dynamic data schema with pre-aggregations separated by tenant

- Dynamically fetch tenants from the database
- Generate separate names for the tenant’s pre-aggregations

Add a pre-aggregation

```
regionCustomerOrderCount: {
  measures: [
    Order.count
  ],
  dimensions: [
    Customer.cName,
    Region.rName,
    Region.rRegionkey
  ],
  timeDimension: Order.oOrderdate,
  granularity: `month`,
  partitionGranularity: `month`,
  refreshKey: {
    every: `1 hour`,
    incremental: true,
    updateWindow: `1 month`,
  },
}
```

Fetch tenants from the DB with BigQuery SDK

```
const { BigQuery } = require('@google-cloud/bigquery');
const bigquery = new BigQuery();

async function fetchRegionKeys() {
  const regionsQuery = `
    SELECT DISTINCT R_REGIONKEY
    FROM \`cube-devrel-team.tpc_h.region\`
  `;

  const options = { query: regionsQuery, location: 'US' };
  const [ job ] = await bigquery.createQueryJob(options);

  const [ rows ] = await job.getQueryResults();
  const regionKeys = rows.map(row => row['R_REGIONKEY']);

  return regionKeys;
};
```

Configure dedicated pre-aggregation schemas

```
contextToAppId: ({ securityContext }) =>
    `CUBEJS_APP_${securityContext.rRegionkey}`,
preAggregationsSchema: ({ securityContext }) =>
    `pre_aggregations_${securityContext.rRegionkey}`,

scheduledRefreshContexts: async () => {
  const rRegionkeys = await fetchRegionKeys();

  function mapSecurityContext() {
    return rRegionkeys.map(rRegionkey => {
      return { securityContext: { rRegionkey } }
    })
  }
  return mapSecurityContext();
}
```

Third-party dependencies in Cube Cloud

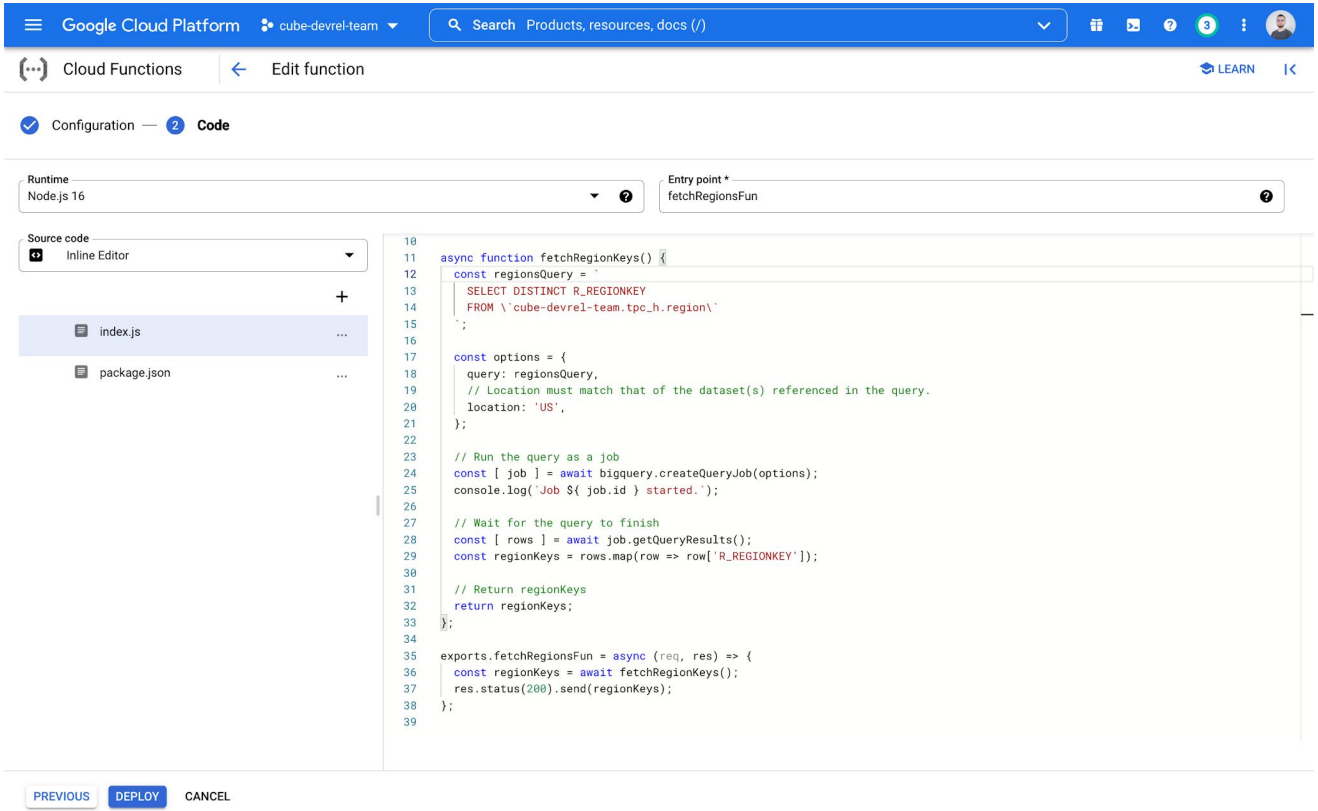
In Cube Cloud

- `require(...)` is disabled
- Contact support if you need 3rd party deps
- Can require `@cubejs-backend/*-driver` packages

Workaround with Serverless Functions

- Use a function to fetch tenants from the database with BigQuery SDK

GCP Function to fetch tenants



`cube.js` config with GCP Function

```
const request = require('./utils/request');
async function fetchRegionKeysGCPFun() {
  const options = {
    host: 'us-central1-cube-devrel-team.cloudfunctions.net',
    path: '/fetchTpchRegions',
    port: '443'
  };
  const regionKeys = await request(options);
  return JSON.parse(regionKeys);
};
```

`request.js` helper

```
const https = require('https');
function request(options) {
  return new Promise((resolve, reject) => {
    https
      .request(options, function (response) {
        let responseBody = '';
        response.on('data', function (data) {
          responseBody += data;
        });
        response.on('end', function () {
          resolve(responseBody);
        });
        response.on('error', function (e) {
          throw e;
        });
      })
      .end();
  });
}
module.exports = request;
```

Demo: Step 9 / Part 2
Dynamic data schema with pre-aggregations separated by tenant

Demo: Step 9 / Part 2

- Add GCP Function in `cube.js` file
- Add dedicated pre-aggregation builds for each tenant
- Build pre-aggregations based on `securityContext`
- View the separate pre-aggregation tables in GCS

Resources

To learn more about pre-aggregations

- Documentation: https://cube.dev/docs/caching
- Blog posts:
  - https://cube.dev/blog/high-performance-data-analytics-with-cubejs-pre-aggregations/
  - https://cube.dev/blog/data-warehouse-performance-and-how-cube-can-help/
- Community support:
  - Discourse: https://forum.cube.dev/
  - Slack: http://cube-js.slack.com/


cube