**Chapter 3 – Infrastructure Optimization**

**Epic Game Store**

**CDOF3 - LE QUERNEC Loévan / MARTIN Santiago / RINGEISSEN Léo / SALEH Julien**

Table of Contents

[MongoDB Queries 2](#_Toc182833164)

[Sharding Key Selection 13](#_Toc182833165)

[Cost Calculation – Network Cost Model 14](#_Toc182833166)

[Conclusion 14](#_Toc182833167)

# MongoDB Queries

## View reviews for a specific game

db.games.aggregate([

  {

    $match: { game\_slug: "celeste" }

  },

  {

    $lookup: {

      from: "open\_critic",

      localField: "id",

      foreignField: "game\_id",

      as: "reviews"

    }

  },

  {

    $project: {

      name: 1,

      "reviews.author": 1,

      "reviews.rating": 1,

      "reviews.comment": 1,

      "reviews.date": 1

    }

  }

])

Une image contenant capture d’écran

Description générée automatiquement

We have here maybe between 20 and 30 critics, average for an independent game.

## Find hardware requirements for a game

db.games.aggregate([

  {

    $match: { game\_slug: "celeste" }

  },

  {

    $lookup: {

      from: "necessary\_hardware",

      localField: "id",

      foreignField: "fk\_game\_id",

      as: "necessaryHardware"

    }

  },

  {

    $project: {

      name: 1,

      "necessaryHardware.operating\_system": 1,

      "necessaryHardware.processor": 1,

      "necessaryHardware.memory": 1,

      "necessaryHardware.graphics": 1,

      "necessaryHardware.graphics\_memory": 1

    }

  }

])

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

In this request, we have two results, because we have the minimum and the recommended hardware for each game. Here, for Celeste, because it does not require a huge configuration, minimum = recommended hardware.

## Search for social networks for a game

db.games.aggregate([

  {

    $match: { game\_slug: "rainbow-six-siege" }

  },

  {

    $lookup: {

      from: "social\_networks",

      localField: "id",

      foreignField: "fk\_game\_id",

      as: "socialNetworks"

    }

  },

  {

    $project: {

      name: 1,

      "socialNetworks.url": 1,

      "socialNetworks.description": 1

    }

  }

])

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Rainbow Six Siege has a few links to different social networks.

## View tweets related to a game

db.games.aggregate([

  {

    $match: { game\_slug: "shadow-complex" }

  },

  {

    $lookup: {

      from: "twitter\_accounts",

      localField: "id",

      foreignField: "fk\_game\_id",

      as: "twitterAccounts"

    }

  },

  {

    $unwind: "$twitterAccounts"

  },

  {

    $lookup: {

      from: "tweets",

      localField: "twitterAccounts.id",

      foreignField: "twitter\_account\_id",

      as: "tweets"

    }

  },

  {

    $unwind: "$tweets"

  },

  {

    $project: {

      name: 1,

      "twitterAccounts.username": 1,

      "tweets.text": 1,

      "tweets.quantity\_likes": 1,

      "tweets.timestamp": 1

    }

  }

]);

Une image contenant texte, capture d’écran

Description générée automatiquement

Here, we searched the tweets related to the game ‘shadow-complex’ and we found some amout of tweets related to it. The amount of tweets related is different for each game, it’s a independent game here, so not a huge amount.

## Average rating for each game over time

db.games.aggregate([

  {

    $lookup: {

      from: "open\_critic",

      localField: "id",

      foreignField: "game\_id",

      as: "reviews"

    }

  },

  {

    $unwind: "$reviews"

  },

  {

    $addFields: {

      "reviews.date": { $toDate: "$reviews.date" }

    }

  },

  {

    $group: {

      \_id: { name: "$name", month: { $month: "$reviews.date" } },

      avgRating: { $avg: "$reviews.rating" },

      reviewCount: { $sum: 1 }

    }

  },

  {

    $project: {

      name: "$\_id.name",

      month: "$\_id.month",

      avgRating: 1,

      reviewCount: 1

    }

  }

])

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

## Identify most active games on twitter

db.games.aggregate([

  {

    $lookup: {

      from: "twitter\_accounts",

      localField: "id",

      foreignField: "fk\_game\_id",

      as: "twitterAccounts"

    }

  },

  {

    $unwind: "$twitterAccounts"

  },

  {

    $lookup: {

      from: "tweets",

      localField: "twitterAccounts.id",

      foreignField: "twitter\_account\_id",

      as: "tweets"

    }

  },

  {

    $unwind: "$tweets"

  },

  {

    $group: {

      \_id: "$name",

      tweetCount: { $sum: 1 }

    }

  },

  {

    $sort: { tweetCount: -1 }

  }

]);



Now we have a tweets count for each game. This query takes a huge time compared to the other queries, because the collection *tweets* has a lot of data.

## Top 5 publishers by average game rating

db.games.aggregate([

  {

    $lookup: {

      from: "open\_critic",

      localField: "id",

      foreignField: "game\_id",

      as: "reviews"

    }

  },

  {

    $unwind: "$reviews"

  },

  {

    $group: {

      \_id: "$publisher",

      avgRating: { $avg: "$reviews.rating" },

      reviewCount: { $count: {} }

    }

  },

  {

    $match: { reviewCount: { $gte: 10 } }

  },

  {

    $sort: { avgRating: -1 }

  },

  {

    $limit: 5

  }

])

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

## Trending genres based on twitter activity and reviews

db.twitter\_accounts.aggregate([

  // we join the tweets with the twitter accounts

  {

    $lookup: {

      from: "tweets",

      localField: "id",

      foreignField: "twitter\_account\_id",

      as: "tweets"

    }

  },

  // we add a field to count the number of tweets for each account

  {

    $addFields: {

      tweetCount: { $size: "$tweets" }

    }

  },

  // we join the twitter accounts with the games

  {

    $lookup: {

      from: "games",

      localField: "fk\_game\_id",

      foreignField: "id",

      as: "games"

    }

  },

  {

    $unwind: "$games"

  },

  {

    $unwind: "$games.genres"

  },

  // we regroup the games by genre

  {

    $group: {

      \_id: "$games.genres",

      totalTweetCount: { $sum: "$tweetCount" }

    }

  },

  // we sort the genres by the total number of tweets

  { $sort: { totalTweetCount: -1 } }

])

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Here, the most difficult query we had to do. It’s different because we aggregate from the *twitter\_accounts* collection to join on the *games* and *tweets* collections. We implemented a counter to finally count the tweets for each genre.

# Sharding Key Selection

## Strategy 1 : ShardinG BY GAME SLUg

**Optimized for user queries : Shard by game\_slug (hashed) and ensure secondary indexing on publisher to handle analytics queries.**

Advantages:

* The relatively small size of the games collection ensures efficient lookup when combined with the game\_slug key.
* Joins with tweets, open\_critic, or other collections remain efficient because the majority of queries start with a filter on game\_slug.

Challenges:

* The large size of the tweets.csv file introduces potential shard imbalance if some games generate significantly more tweets than others (e.g., trending games).

## Strategy 2 : ShardinG BY publisher

**Optimized for analytics : Shard by publisher (hashed), ensuring efficient aggregation and secondary indexes on game\_slug for user lookups.**

Advantages:

* Distributes the large tweets and open\_critic data across publishers.
* Ideal for queries aggregating data by publisher or across genres, as this avoids scatter-gather across shards.

Challenges:

* Scatter-gather queries will be frequent for user-facing queries filtering by game\_slug.

# Cost Calculation – Network Cost Model

Here is our cost model:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Query | Database Schema 1 | | Database Schema 2 | |
| Sharding 1 Cost | Sharding 2 Cost | Sharding 1 Cost | Sharding 2 Cost |
| R𝑢1 | 10 | 30 | 50 | 80 |
| R𝑢2 | 10 | 25 | 40 | 70 |
| R𝑢3 | 10 | 20 | 50 | 80 |
| R𝑢4 | 30 | 50 | 70 | 100 |
| R𝑑𝑎1 | 60 | 100 | 40 | 70 |
| R𝑑𝑎2 | 70 | 120 | 50 | 80 |
| R𝑑𝑎3 | 90 | 150 | 70 | 110 |
| R𝑑𝑎4 | 100 | 180 | 80 | 140 |
| Weighted Total | 123580 | 349860 | 576110 | 926880 |

# Conclusion

Here are the many observations we can make from the board we got in the previous section.

Schema 1's Overall Efficiency:

* Schema 1 consistently shows lower computational costs across most queries, suggesting it is inherently more efficient due to better data organization for general-purpose workloads.
* This makes Schema 1 better suited for mixed-use scenarios, even though it is primarily optimized for user-facing queries.

Sharding 2's Performance for Analytical Queries:

* Sharding 2 performs better for analytical queries on Schema 2, as the shard distribution aligns well with aggregation-heavy workloads.
* However, its user query performance is significantly worse on Schema 2, leading to a higher overall weighted cost despite its analytical efficiency.

Schema 2's Strengths and Weaknesses:

* Schema 2 shows improved performance for analytical queries when mixed with Sharding 2 but is worse from poor user-query performance with both sharding strategies.
* This trade-off emphasizes that Schema 2 is better tailored for analytical workloads, but its structure incurs losses for user-facing queries.

Sharding-Schema Alignment:

* Schema 1 + Sharding 1 is the most balanced and efficient pairing, excelling in user-facing queries while maintaining acceptable performance for analytical queries.
* Schema 2 + Sharding 2 delivers the best performance for analytical queries but at the expense of user query efficiency, leading to the worst overall weighted cost.