SnipMan

Organize your Code Snippets

Group 42

Péter Ferenc Gyarmati 11913346

Simon Eckerstorfer 11911424

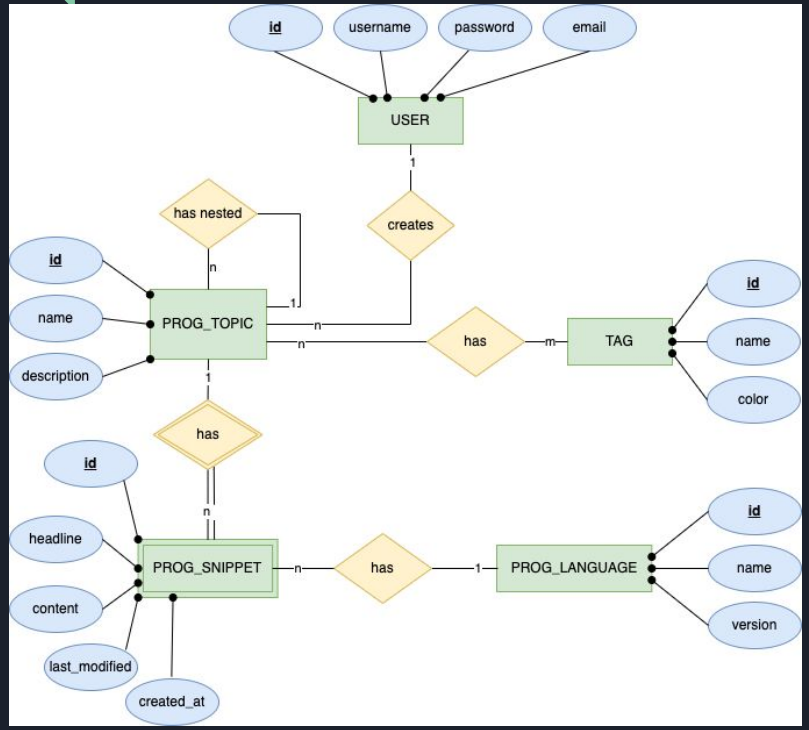
Overview

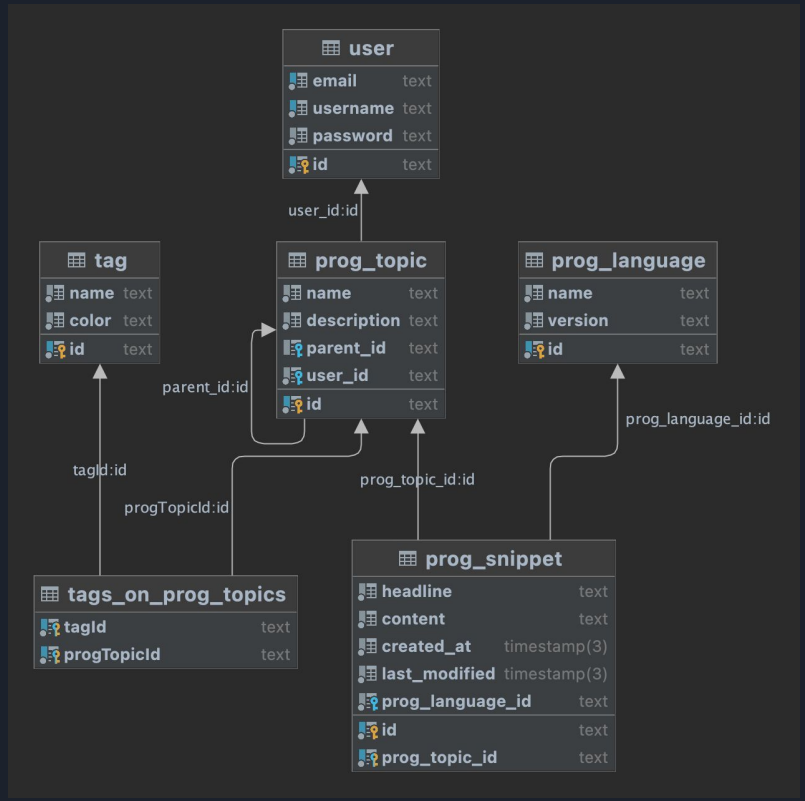
Comparison of RDBMS model and NoSQL Design

Performance Implications of NoSql for Reports

Query Comparison

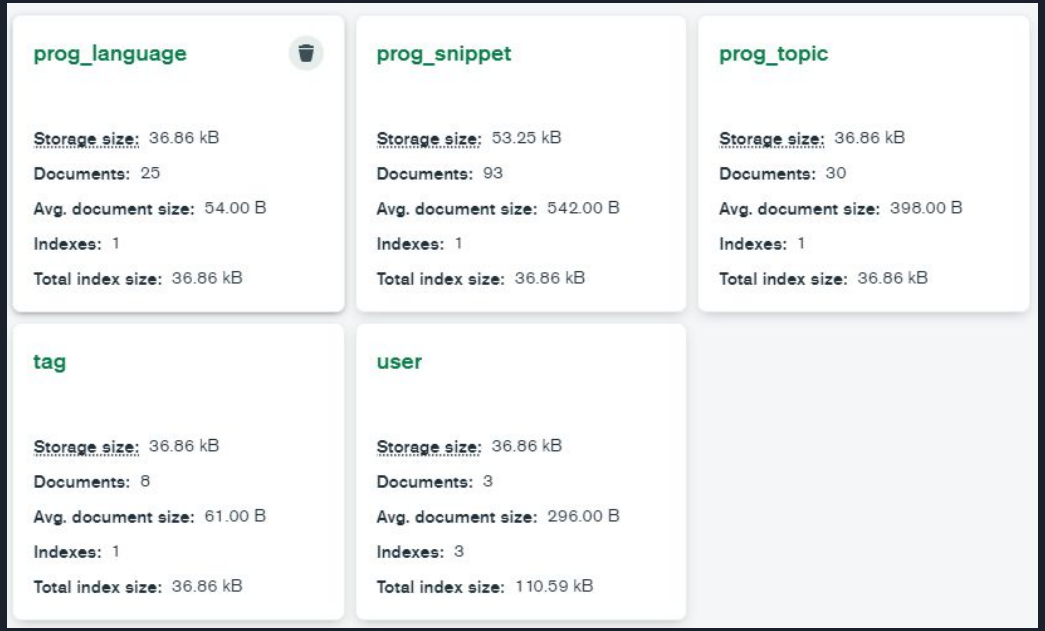
Comparison of **RDBMS model** and NoSQL Design





Comparison of RDBMS model and **NoSQL Design**

- One collection for each data model



Comparison of RDBMS model and **NoSQL Design**

- **User** collection
- Topics big and nested -> only IDs

```
_id: ObjectId('62ab09fb89987f389fc218c5')
email: "Laverne_Carroll71@hotmail.com"
username: "Laverne_Carroll71"
password: "Kp4qWYmHlf0zlkG"
prog_topic_ids: Array
  0: ObjectId('62ab09fb89987f389fc218c8')
  1: ObjectId('62ab09fb89987f389fc218c9')
```

- **Tag** collection for efficient queries
- Tiny documents, mostly read with topics, rarely writen
  - also embedded in topics (see next slide)

```
_id: ObjectId('62ab09fc89987f389fc218e6')
name: "svelte-components"
color: "#18771a"
```

Comparison of RDBMS model and **NoSQL Design**

- **Topics** collection
- Snippets not limited in size, may get very big -> only IDs
- Nesting depth of topics not limited -> only parent ID
  - No parent -> root topic
- Tags needed on every topic read -> embedded

Comparison of RDBMS model and **NoSQL Design**

- **Snippet** collection
- Weak entity, but may get big -> related topic ID saved
- Programming Languages are tiny + needed on most reads -> embedded
- User Email needed for report -> included in document

- **Programming** Languages similar to Tags - tiny documents
  - Own collection for efficient query
  - Embedded in snippet

Comparison of **RDBMS model** and **NoSQL Design**

- More flexible data models using NoSQL
- The NoSQL design made it possible to customize the stored documents' structure to be ideal for our exact use cases
- Data that is accessed together is stored together (mostly)

Performance Implications of NoSQL for Reports

Active Users Report (Péter)

- Programming Language and User Email address embedded
- Projection and filtering before data joining
- TTL Index for *prog_snippet.created_at*

Dominant Languages Report (Simon)

- Programming Languages are embedded -> no join necessary
- Lookup still needed for snippets (Snippets not embedded on Topics)
- Projecting and matching
    - In particular before lookup

Query Comparison - **Active Users**

No joins (lookups) are necessary, as we embedded `user_email` and `prog_language` into snippets

Joins of `prog_topic`, `relevant_snippets` and user are necessary, to access `user_email`

```sql
--- Initial filtering for snippets with given lang and timestamp
WITH relevant_snippets AS (
    SELECT id, prog_topic_id
    FROM prog_snippet
    WHERE prog_language_id = ${progLanguageId}
    AND created_at BETWEEN (now() - INTERVAL '1 month') AND now()
),
--- After filter, perform join and aggregation to access the relevant user info
report_result AS (
    SELECT "user".email AS user_email
    FROM prog_topic
    JOIN "user" ON "user".id = prog_topic.user_id
    JOIN relevant_snippets ON relevant_snippets.prog_topic_id = prog_topic.id
    GROUP BY user_email
    HAVING COUNT(relevant_snippets.id) >= 3
)
--- Order the result set by the email address
SELECT user_email
FROM report_result
ORDER BY user_email;
```

```javascript
db.prog_snippet.aggregate([
  // 1. Keep only the relevant fields, get the creation date as ISO string
  {
    $project: {
      _id: 1,
      prog_language_name: "$prog_language.name",
      prog_snippet_created_at: {
        $convert: { input: "$created_at", to: "string" }
      },
      user_email: 1
    }
  },
  // 2. Keep only the relevant snippets
  {
    $match: {
      $expr: {
        $and: [
          { $eq: ["$prog_language_name", progLanguageName] },
          { $gte: ["$prog_snippet_created_at", oneMonthAgo] },
          { $lte: ["$prog_snippet_created_at", now] }
        ]
      }
    }
  },
  // 3. Group by email, assign it to doc _id, as it is unique at this stage
  { $group: { _id: "$user_email", count: { $sum: 1 } } },
  // 4. Keep only those emails where the snippet count is at least 3
  { $match: { $expr: { $gte: ["$count", 3] } } },
  // 5. Keep only the email field, without the count
  { $project: { _id: 0, email: "$_id" } },
  // 6. Sort by email
  { $sort: { email: 1 } }
]);
```

Query Comparison - **Dominant Languages**

```
--- 2. Select snippets that are in relevant topics
--- and create a column for the length of their content
WITH relevant_snippets AS (
    SELECT prog_language_id, LENGTH(content) as local_len
    FROM prog_snippet
    WHERE prog_topic_id IN
--- 1. Select relevant topics for given tag id
        (SELECT progTopicId
        FROM tags_on_prog_topics
        WHERE tagId = ${tagId}))
--- 3. Join with prog language and group snippets by language
--- create an aggregate column with the sum of the grouped snippets
--- Order by the length and take the first 10
SELECT name, version, SUM(local_len) as length
FROM relevant_snippets
JOIN prog_language
    ON prog_language.id = relevant_snippets.prog_language_id
GROUP BY prog_language_id, name, version
ORDER BY length DESC
LIMIT 10
```

- **SQL**: Join with prog_language necessary to obtain name & version
- **NoSQL**: NOT necessary (embedded)
- Lookup of prog_snippet still needed in **NoSQL** (too large to embed)

Query Comparison - **Dominant Languages**

```
db.progTopic.aggregate([
  {
    // 1 .filter topics with the given tag name
    $match: {
      tags: {
        $elemMatch: { name: tagName },
      },
    },
  },
  {
    // 2. only keep snippet id arrays
    $project: {
      _id: 0,
      prog_snippet_ids: 1,
    },
  },
  {
    // 3. join snippets that are associated with the topics
    // as prog_snippet_ids is an array, prog_snippets is also an array (of snippets)
    $lookup: {
      from: 'prog_snippet',
      localField: 'prog_snippet_ids',
      foreignField: '_id',
      as: 'prog_snippets',
    },
  },
  {
    // 4. only keep snippet arrays
    $project: {
      _id: 0,
      prog_snippets: 1,
    },
  },
  {
    // 5. create a document for each snippet
    $unwind: '$prog_snippets',
  },
  {
    // 6. keep only the relevant fields
    $project: {
      lang: '$prog_snippets.prog_language',
      content: '$prog_snippets.content',
    },
  },
  {
    // 7. group by language
    // create an aggregate length field with the sum of the length of the content
    $group: {
      _id: '$lang',
      length: {
        $sum: { $strLenCP: '$content' },
      },
    },
  },
  {
    // 8. map fields to create wanted output format
    $project: {
      _id: 0,
      name: '$_id.name',
      version: '$_id.version',
      length: 1,
    },
  },
  {
    // 9. sort by length descending and name ascending
    $sort: {
      length: -1,
      name: 1,
    },
  },
  {
    // 10. only take the first 10 results
    $limit: 10,
  },
],);
```

Thank You
for Your Attention