Relational Database Performance for Multimedia: A Case Study

Björn Þór Jónsson*
bjorn@ru.is
Reykjavik University
Reykjavík, Iceland

Aaron Duane
aadu@itu.dk
IT University of Copenhagen
Copenhagen, Denmark

Nikolaj Mertz
nime@itu.dk
IT University of Copenhagen
Copenhagen, Denmark

ABSTRACT

This paper describes the performance optimisation of a state-of-the-art relational database to more efficiently serve data for multimedia visualisations in the ViRMA prototype. We describe the baseline database and queries, along with two major optimisation steps that improve query efficiency, at the cost of slowing down dynamic updates. We evaluate the optimisations with a case study of a lifelog collection of 182K images, showing that the time to produce complex visualisations is reduced by orders of magnitude.

CCS CONCEPTS

• **Information systems** → **Multimedia and multimodal retrieval**; **Multimedia databases**.

KEYWORDS

Performance tuning, relational database, ViRMA

ACM Reference Format:
Björn Þór Jónsson, Aaron Duane, and Nikolaj Mertz. 2022. Relational Database Performance for Multimedia: A Case Study. In *International Conference on Content-based Multimedia Indexing (CBMI 2022), September 14–16, 2022, Graz, Austria.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3549555.3549558

1 INTRODUCTION

As multimedia collections grow in both importance and scale, so grows the need to run complex queries and tasks against these collections. Historically, much emphasis has been placed on similarity search, but recently an argument has been made that more interactive and exploratory approaches would serve users better [14, 18]. Such applications must eventually be served by a database system of some sort with the necessary query processing capabilities. Given the prevalence of relational database know-how in industry and academia, and the ability of relational systems to optimise and serve complex queries in other contexts, it is a tempting first step to employ relational systems to the extent possible. It is well known that relational systems support high-dimensional indexing poorly, if at all, but surely they must work for some of the metadata queries we might like to answer. The question considered here is: how well?

*Research conducted while the author was with the IT University of Copenhagen.

CBMI 2022, September 14–16, 2022, Graz, Austria

ACM ISBN 978-1-4503-9720-9/22/09…$15.00
https://doi.org/10.1145/3549555.3549558

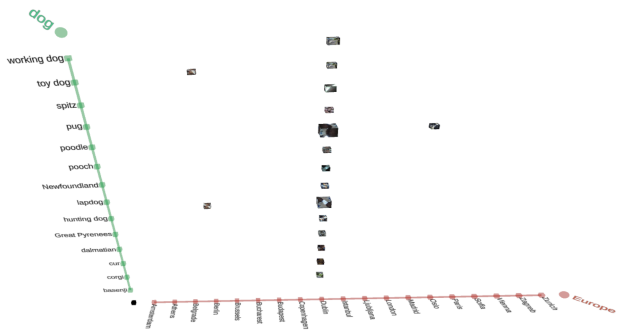

Figure 1: Browsing state visualisation in ViRMA, with axes showing European time zones and different dog breeds.

The Multidimensional Media Model (M^3, pronounced "emm-cube") is an example of a metadata-based approach to serving multimedia applications [9]. In its first incarnation, M^3 is a relatively direct adaptation of OLAP technologies to multimedia, which considers multimedia to reside in a multi-dimensional metadata space, and uses a variant of faceted search to (a) filter the metadata space and (b) project it to a 3D visual representation that can explored by the user (Figure 1). The M^3 model can be easily described using an ER-diagram and translated into relational tables, and queries to project the metadata space to the visual representation are conceptually simple. The ViRMA system [5], which is the latest implementation of the M^3 model, therefore queries a relational system.

Running complex interactions efficiently, however, is not a trivial task. In this paper, we describe the process of optimising a state-of-the-art relational database to serve visualisations for the ViRMA system. We outline the steps taken to denormalise tables and rewrite queries, and analyse the resulting query plans. Through these optimisation steps, the time to produce complex visualisations is improved by orders of magnitude in a lifelog collection of 182K images, 107K distinct tags, and 6.3M image-tag associations. The results indicate that even for complex visualisations, performance is sufficient for interactive exploration at this scale. The performance improvements, however, come at a significant cost of updates to the collection. We therefore conclude that the state-of-the-art relational server is overall not suitable for the ViRMA system.

2 BACKGROUND

In this section, we first present the LSC collection which is the topic of our case study. We then outline the M^3 model [9], and describe the database implementation of the M^3 model in the ViRMA prototype.

Case Study: Lifelog Search Challenge. *The Lifelog Search Challenge (LSC) is an annual live event for media retrieval systems. Before the competition, an image collection is distributed to competitors,*

along with its metadata. The teams can then augment the metadata with analysis methods of their choice. During the competition, tasks are given as text descriptions that are gradually disclosed, and the teams have 5-7 minutes to find one of the images that are considered ground truth to the tasks. An example task, that we use as a running example in this paper, involves looking for images from a weekend garden party with friends in Dublin, where there was a dog. The score for each task is based on the time needed to solve it and the number of incorrect attempts, and scores of teams are then aggregated across tasks to determine the winner. A variety of systems compete in LSC each year [7], focusing on a variety of approaches, including event detection [16], multimodal retrieval [8, 12], temporal queries [11], relevance feedback [10, 11], and VR interfaces [6, 15].

Brief Overview of the M^3 Model. Several approaches from the literature have explored media collection in a multi-dimensional manner [1–4, 17]. The foundation of the M^3 model is media metadata, which is used to define the dimensions of a *hypercube*, a multidimensional space that organises media items into related groups. Browsing and drilling down into these dimensions allows the user to explore media collections by interactively defining the set of relevant images and visualising them on screen.

In the model, media items are referred to as *objects*, metadata items as *tags*, and the association of a tag to an object as a *tagging*. In an effort to organise media items into groups, distinct tags are arranged into *tagsets*. Tagsets can either be generated manually or derived from existing structures in the metadata. *Hierarchies* are then defined over tagsets to add structure to their tags. Tagsets and hierarchies together form the *dimensions* of the media hypercube.

Given the large number of potentially interesting dimensions of media metadata, the hypercube is a conceptual construct that can only be visualised by projecting some of its dimensions to a 1D, 2D or 3D *exploration cube*. Figure 1 shows an example of a 2D visualisation, described in more detail below. Exploring a collection thus requires the user to decide which of the available dimensions should be visualised on each of the three available axes of the exploration cube at any given time. Since the content of the exploration cube changes dynamically, as the user makes new decisions on which dimensions to project, we refer to the structure of the exploration cube at each time as the *browsing state*. Each projected dimension has a set of tags on its axis, which are either the tags of a tagset or the children of one node of a hierarchy. The combinations of these tags then form *cells*, where each cell represents the images associated with that combination of tags.

Projecting a hierarchy dimension allows the user to traverse up and down the chosen hierarchy, always projecting the children of one node to the exploration cube. These actions, referred to as *rolling up* or *drilling down*, directly correspond to applying a *hierarchy filter* on the parent node at each time, limiting the images presented in each cell to those associated with any of the tags present in the corresponding subtree. Likewise, projecting a tagset corresponds to applying a *tagset filter*, which restricts the visible images to those associated with one or more of the tags in the tagset. The user can dynamically decide to change the projected dimension on one of the axes, which is referred to as *pivoting*. During pivoting, any filters applied to the replaced dimension are maintained, and a new tagset or hierarchy filter is applied to the newly projected

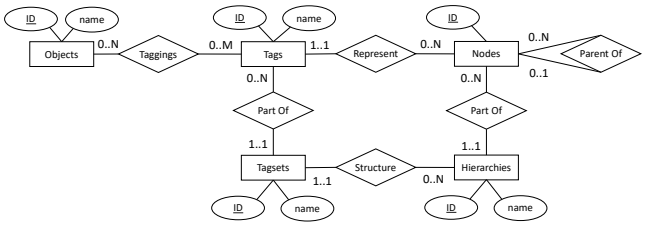

Figure 2: Simplified ER diagram for the M^3 model.

```
CREATE FUNCTION get_subtree_from_parent_node(INT)
RETURNS SETOF Nodes
AS $$
WITH RECURSIVE all_sub_nodes AS (
  SELECT N.id, N.tag_id, N.hierarchy_id, N.parentnode_id
  FROM Nodes N
  WHERE N.id = $1
  UNION ALL
  SELECT N.id, N.tag_id, N.hierarchy_id, N.parentnode_id
  FROM Nodes N JOIN all_sub_nodes A ON A.id = N.parentnode_id
) SELECT * FROM all_sub_nodes
$$ LANGUAGE SQL;
```

Listing 1: Recursive query for the subtree of a parent node.

dimension. Furthermore, *tag filters*, requiring association with a specific tag, and *range filters* over some tagsets can be used to slice the collection even further without requiring projection to an axis.

During exploration, the user can apply a multitude of filters, and dynamically decide which dimensions to project to the exploration cube, thus continually updating the current browsing state.

Case Study: Lifelog Database. *From the metadata collection associated with the LSC collection, we have extracted tags for each image into 10 different tagsets, of which the most important are date, time, time zone, and location. From the image date and time of creation, we have then derived tags into 9 tagsets with more details, such as day of week, day of month, month, year, and hour. Finally, we have generated an entity tagset, extracted using ImageNet Shuffle [13]. This tagset contains the semantics tags (e.g.,* Corgi *and* Coffee table*) used to describe the visual contents of each image. For each of the images, all relevant concepts are retained as tags, resulting in 9,456 different tags. Finally, we created a hierarchy for the concept tagset using the WordNet Python API.*

Consider the example task of looking for images from a weekend garden party with friends in Dublin, where there was a dog. As a starting point, the user might apply filters to only show images where the day of week tag is Saturday or Sunday. The user might then create a visualisation with all children of the Dog node from the entity hierarchy on one axis, and the time zone hierarchy on another axis. The resulting visualisation, with the different dog types on one axis and time zones in Europe on the other, only showing images taken on weekends, is the one shown in Figure 1. The user can further interact with the collection, e.g., drilling deeper into the semantic label hierarchy, adding a filter to focus on images from Dublin, or projecting another tagset or hierarchy node to the third visual axis.

The ViRMA Database. Figure 2 shows a simplified ER-diagram for the M^3 model. As the figure shows, there are entities for Objects

```
SELECT S.idx AS x, S.idy AS y, S.idz AS z, S.object_id AS id, O.file_uri AS fileURI, S.cnt AS cnt
FROM (SELECT R1.id AS idx, R2.id AS idy, 1 AS idz, MAX(R1.object_id) AS object_id, COUNT(DISTINCT R1.object_id) AS cnt
      FROM (SELECT R.object_id, H.id AS id
            FROM (SELECT N.parentnode_id, N.id, (get_subtree_from_parent_node(N.id)).tag_id FROM Nodes N) H
            JOIN Taggings R ON R.tag_id = H.tag_id
            WHERE H.parentnode_id = 732) R1
      JOIN (SELECT R.object_id, R.tag_id AS id
            FROM Tags T
            JOIN Taggings R ON R.tag_id = T.id
            WHERE T.tagset_id = 13) R2 ON R1.object_id = R2.object_id
      JOIN (SELECT R.object_id FROM Taggings R WHERE R.tag_id IN (10567,22)) R3 ON R1.object_id = R3.object_id
      GROUP BY idx, idy, idz) S
JOIN Objects O ON X.object_id = O.id;
```

Listing 2: Baseline query for returning the browsing state of Figure 1.

```
CREATE MATERIALIZED VIEW Nodes_Taggings AS
SELECT H.parentnode_id, H.id AS node_id, H.tag_id, R.object_id
FROM (SELECT N.parentnode_id, N.id, (get_subtree_from_parent_node(N.id)).tag_id FROM Nodes N) H
JOIN Taggings R ON R.tag_id = H.tag_id;
```

Listing 3: Materialised view that stores flattened hierarchies, along with related tags and objects.

```
CREATE MATERIALIZED VIEW tagsets_taggings AS
SELECT T.tagset_id AS tagset_id, R.tag_id, R.object_id
FROM Tags T
JOIN Taggings R ON R.tag_id = T.id;
```

Listing 4: Materialised view that stores relations between tagsets and objects.

```
SELECT S.idx AS x, S.idy AS y, S.idz AS z, S.object_id AS id, O.file_uri AS fileURI, S.cnt AS cnt
FROM (SELECT R1.id AS idx, R2.id AS idy, 1 AS idz, MAX(R1.object_id) AS object_id, COUNT(DISTINCT R1.object_id) AS cnt
      FROM (SELECT N.object_id, N.node_id AS id FROM Nodes_Taggings N WHERE N.parentnode_id = 732) R1
      JOIN (SELECT T.object_id, T.tag_id AS id FROM Tagsets_Taggings T WHERE T.tagset_id = 13) R2  ON R1.object_id = R2.object_id
      JOIN (SELECT R.object_id FROM Taggings R WHERE R.tag_id IN (10567,22)) R3 ON R1.object_id = R3.object_id
      GROUP BY idx, idy, idz) S
JOIN Objects O ON S.object_id = O.id;
```

Listing 5: Optimised baseline query for the browsing state of Figure 1, rewritten to use the materialised views.

(here, images), Tagsets, Tags, Hierarchies and Nodes, along with a many-to-many relationship for Taggings, resulting in a textbook mapping to 6 relations with the appropriate `PRIMARY KEY` and `FOREIGN KEY` declarations. We note that since the Nodes relation implements the actual hierarchies, the recursive query in Listing 1 is needed to extract information about subtrees of parent nodes.

Case Study: Lifelog Browsing State. *Listing 2 shows a baseline query to return a variant of the browsing state in Figure 1. This query has one subquery for each projected dimension and each direct filter. For the hierarchy projection of the Dog node, the first subquery returns all objects tagged to the subtree of the parent node, along with identifiers of the relevant child nodes used to locate the image in the browsing state visualisation. On the second axis we have introduced a time zone tagset projection; the second subquery returns objects tagged with any tag from the tagset, along with the identifiers of all associated tags. The last subquery applies the weekday filter, returning only images taken on weekends. The outer query then joins the subqueries and groups the resulting images into browsing state cells, where they are counted and an image chosen as a representative. Finally, the URLs of the representative images are obtained from the Objects table.*

Note that as the user progresses in the interactive exploration, a query like the one shown in Listing 2 must be issued for each browsing state. The costliest operation in this baseline query is clearly the recursive query to identify all the nodes in a subtree. In the following section, we therefore explore the available options to reduce the cost of this operation and tune the performance

3 TUNING STEPS

In this section, we describe the steps taken to improve the performance of the baseline cell query from the previous section. We first describe the optimisation of the existing relations, then present a denormalisation approach to reduce/remove the cost of dynamically computing subtrees, and finally describe indexes created on the materialised views that result in more efficient execution.

Step 0: Indexes and Statistics. To facilitate efficient computation of subtrees for the baseline query, we created a clustered index on the Nodes(parentnode_id) attribute, which groups together all children of each node. We also created an index on Taggings(object_id, tag_id) to facilitate efficient computation of tag filters. Finally, we analysed statistics for the collection, using default settings, to improve query optimisation. These basic steps were taken before measuring the performance of the baseline query of Listing 2.

Step 1: Denormalisation. The main cost of the baseline query is the recursive query to get all the tags represented in the subtree of a node. The first step of performance tuning is therefore to *precompute*, using a materialised view, the subtrees of all nodes and their tagging information, thus *denormalising* the database. The view of Listing 3 essentially results in a new table with about 32M

Table 1: Description of browsing states used in experiments.

State	Description
2D	2D browsing state of 39 cells, with the top level of the dog hierarchy on one axis and year on the other axis.
3D	3D browsing state of 4,186 cells, with the top level of the dog hierarchy on the first axis, location on the second axis, and day of the week on the third axis.
3D + 2	3D browsing state of 966 cells, with location on the first axis, day of the week on the second axis, year on the third axis, and filters on dog and the month of September.

rows. Similarly, in Listing 4, a materialised view is created to store relationships between tagsets and objects, resulting in table with about 6.3M rows. Computing these views takes nearly 60 seconds.

Listing 5 shows the baseline query rewritten to take advantage of the materialised views. As each view essentially implements the subqueries of the baseline query in Listing 2, the resulting optimised query is significantly simpler in its structure. Furthermore, this optimised query runs about 75% faster than the baseline query, as detailed in Section 4.

Step 2: Covering Indexes. We observe that every dimension subquery requires three columns, in this order: (i) parentnode_id/tagset_id to filter the relevant parent node or tagset; (ii) object_id to join with the other dimensions/filters; and (iii) node_id/tag_id for grouping the objects based on their location in the browsing state. To facilitate the queries, we have therefore created two *covering* B+-tree indexes for the two views, as well as a third index for implementing direct hierarchy filters. Note that this action does not necessitate any further changes to the browsing state query in Listing 5. By adding these indexes, the time to retrieve the example browsing state is further reduced by more than 90%.

4 EVALUATION

4.1 Experimental Setup

To measure the impact of the tuning actions of the previous section, we use the three browsing states described in Table 1. The three browsing states represent a range of browsing state complexities, from a relatively simple 2D state to a very complex 3D state with additional filters. For the more complex browsing states, cells tend to have few images, which requires the baseline approach and its variants to process a larger portion of the objects table. To avoid software overhead, a Python benchmarking harness was developed. The mean latency of each state query is reported as the average of 30 runs. Experiments were run on a laptop using Ubuntu LTS on Windows 11, with an Intel i5-8600K 3.6GHz CPU and 8GB of RAM.

4.2 Results

Figure 3 shows the results of the performance measurements. The different versions of baseline and tuned queries are on the x-axis, while the y-axis shows the mean running time of retrieving the entire browsing state for each query formulation. Note the logarithmic scale of the y-axis, chosen because there is a wide range from the fastest to the slowest browsing states.

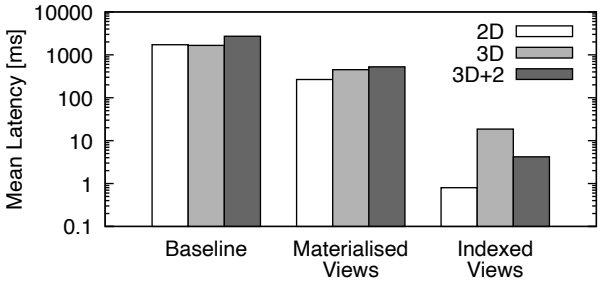

Figure 3: Performance comparison of baseline and optimised queries, over three different browsing state scenarios.

Overall, Figure 3 shows that denormalising the database has a very significant effect on the performance, improving performance by nearly an order of magnitude compared to the baseline approach. Furthermore, the introduction of indexes on the materialised views improves performance by another two orders of magnitude. We note that the additional filter on the month of September reduces the result size of the join for the 3D+2 state, compared to the 3D state, resulting in faster execution.

4.3 Discussion

The results indicate that denormalising and indexing the collection and modifying the baseline query correspondingly is an excellent strategy for visualisation queries. These performance improvements, however, required building a large materialised view, which took nearly a minute to populate and index. Each time the collection is updated by adding an image, a tag, or even a tagging, which will happen frequently in many multimedia analytics scenarios, this view must be updated. For bulk inserts, the view might be refreshed only at the end, but nevertheless this is a significant hindrance to interactive use of such a system to maintain a media collection. We believe that, taking all these considerations together, the relational back-end must be replaced by a more suitable approach.

5 CONCLUSIONS

This paper has described the process of optimising the performance of a state-of-the-art relational database to more efficiently serve data for the visualisation of browsing states in the ViRMA system. Through two major optimisation steps, denormalising the database and creating covering indexes, the time to produce complex browsing states in a lifelog collection of about 182K images was reduced by orders of magnitude. The results show that performance for even complex visualisations is sufficient for interactive exploration. The performance improvements come at the cost of practically prohibiting dynamic updates to the collection, however, and we therefore conclude that even with this relatively simple data model and this relatively small media collection, the state-of-the-art relational server is not suitable technology and must be replaced.

ACKNOWLEDGMENTS

This work was partly supported by MCSA-IF grant 893914.

REFERENCES

[1] Anne-Muriel Arigon, Maryvonne Miquel, and Anne Tchounikine. 2007. Multimedia Data Warehouses: A Multiversion Model and a Medical Application. *Multimedia Tools and Applications* 35, 1 (2007), 91–108.

[2] Ilaria Bartolini and Paolo Ciaccia. 2009. Integrating Semantic and Visual Facets for Browsing Digital Photo Collections. In *Proceedings of the Italian Symposium on Advanced Database Systems (SEBD)*. Camogli, Italy, 65–72.

[3] Ork de Rooij, Cees G. M. Snoek, and Marcel Worring. 2009. MediaMill: Guiding the User to Results Using the ForkBrowser. In *Proceedings of the ACM International Conference on Image and Video Retrieval (CIVR)*. ACM, Santorini, Greece.

[4] Mamadou Diao, Sougata Mukherjea, Nitendra Rajput, and Kundan Srivastava. 2010. Faceted Search and Browsing of Audio Content on Spoken Web. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*. ACM, Toronto, ON, Canada, 1029–1038.

[5] Aaron Duane and Björn Þór Jónsson. 2021. ViRMA: Virtual Reality Multimedia Analytics at LSC 2021. In *Proceedings of the Annual Workshop on Lifelog Search Challenge (LSC@ICMR)*. ACM, Taipei, Taiwan, 29–34.

[6] Aaron Duane, Björn Þór Jónsson, and Cathal Gurrin. 2020. VRLE: Lifelog Interaction Prototype in Virtual Reality: Lifelog Search Challenge at ACM ICMR 2020. In *Proceedings of the Annual Workshop on Lifelog Search Challenge (LSC@ICMR)*. ACM, Dublin, Ireland, 7–12.

[7] Cathal Gurrin, Klaus Schoeffmann, Hideo Joho, Andreas Leibetseder, Liting Zhou, Aaron Duane, Duc-Tien Dang-Nguyen, Michael Riegler, Luca Piras, Minh-Triet Tran, Jakub Lokoč, and Wolfgang Hürst. 2019. Comparing Approaches to Interactive Lifelog Search at the Lifelog Search Challenge (LSC2018). *ITE Transactions on Media Technology and Applications* 7, 2 (2019), 46–59.

[8] Silvan Heller, Mahnaz Amiri Parian, Ralph Gasser, Loris Sauter, and Heiko Schuldt. 2020. Interactive Lifelog Retrieval with vitrivr. In *Proceedings of the Annual Workshop on Lifelog Search Challenge (LSC@ICMR)*. ACM, Dublin Ireland, 1–6.

[9] Björn Þór Jónsson, Grímur Tómasson, Hlynur Sigurþórsson, Áslaug Eiríksdóttir, Laurent Amsaleg, and Marta Kristín Lárusdóttir. 2015. A Multi-Dimensional Data Model for Personal Photo Browsing. In *Proceedings of the International Conference on MultiMedia Modeling (MMM)*. Springer International Publishing, Sydney, NSW, Australia, 345–356.

[10] Omar Shahbaz Khan, Mathias Dybkjær Larsen, Liam Alex Sonto Poulsen, Björn Þór Jónsson, Jan Zahálka, Stevan Rudinac, Dennis Koelma, and Marcel Worring. 2020. Exquisitor at the Lifelog Search Challenge 2020. In *Proceedings of the Annual Workshop on Lifelog Search Challenge (LSC@ICMR)*. ACM, Dublin, Ireland, 19–22.

[11] Gregor Kovalčík, Vít Škrhak, Tomáš Souček, and Jakub Lokoč. 2020. VIRET Tool with Advanced Visual Browsing and Feedback. In *Proceedings of the Annual Workshop on Lifelog Search Challenge (LSC@ICMR)*. ACM, Dublin, Ireland, 63–66.

[12] František Mejzlík, Patrik Veselý, Miroslav Kratochvíl, Tomáš Souček, and Jakub Lokoč. 2020. SOMHunter for Lifelog Search. In *Proceedings of the Annual Workshop on Lifelog Search Challenge (LSC@ICMR)*. ACM, Dublin Ireland, 73–75.

[13] Pascal Mettes, Dennis C. Koelma, and Cees G.M. Snoek. 2016. The ImageNet Shuffle: Reorganized Pre-training for Video Event Detection. In *Proceedings of the ACM International Conference on Multimedia Retrieval (ICMR)*. ACM, New York, NY, USA, 175–182.

[14] Daniel Seebacher, Johannes Häußler, Manuel Stein, Halldor Janetzko, Tobias Schreck, and Daniel A. Keim. 2017. Visual Analytics and Similarity Search: Concepts and Challenges for Effective Retrieval Considering Users, Tasks, and Data. In *Proceedings of the International Conference in Similarity Search and Applications (SISAP)*. Springer, Münich, Germany, 324–332.

[15] Florian Spiess, Ralph Gasser, Silvan Heller, Luca Rossetto, Loris Sauter, Milan van Zanten, and Heiko Schuldt. 2021. Exploring intuitive lifelog retrieval and interaction modes in virtual reality with vitrivr-VR. In *Proceedings of the Annual Workshop on Lifelog Search Challenge (LSC@ICMR)*. ACM, Taipei, Taiwan, 17–22.

[16] Ly-Duyen Tran, Manh-Duy Nguyen, Nguyen Thanh Binh, Hyowon Lee, and Cathal Gurrin. 2020. Myscéal: An Experimental Interactive Lifelog Retrieval System for LSC'20. In *Proceedings of the Annual Workshop on Lifelog Search Challenge (LSC@ICMR)*. ACM, Dublin Ireland, 23–28.

[17] Marcel Worring, Dennis Koelma, and Jan Zahálka. 2016. Multimedia Pivot Tables for Multimedia Analytics on Image Collections. *IEEE Transactions on Multimedia* 18, 11 (2016), 2217–2227.

[18] Jan Zahálka and Marcel Worring. 2014. Towards Interactive, Intelligent, and Integrated Multimedia Analytics. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE, Paris, France, 3–12.