Data Management for Dynamic Multimedia Analytics and Retrieval

Inauguraldissertation

zur

Erlangung der Würde eines Doktors der Philosophie

vorgelegt der

Philosophisch-Naturwissenschaftlichen Fakultät

der Universität Basel

von

Ralph Marc Philipp Gasser

Basel, 2022

Zusammenfassung

Abstract

Acknowledgements

Good luck.

This work was partly supported by the Swiss National Science Foundation, which is also thankfully acknowledged.

Contents

List of Figures

List of Tables

List of Acronyms

1NF	First Normal Form
2NF	Second Normal Form
3NF	Third Normal Form
ACID	Atomicity, Consistency, Isolation, Durability
BCNF	Boyce–Codd Normal Form
CWA	Closed-world assumption
DBMS	Database Management System
DDL	Data Definition Language
DFC	Distance Function Class
DML	Data Manipulation Language
DQL	Data Query Language
ERM	Entity Relationship Model
FK	Foreign Key
FNS	Farthest Neighbour Search
kFN	k-Nearest Neighbour Search
kNN	k-Farthest Neighbour Search
MIPS	Maximum Inner Product Search
MRF	Magnetic Resonance Fingerprinting
NNS	Nearest Neighbour Search
PK	Primary Key
SQL	Structured Query Language

UDF Used Defined Function

List of Symbols

Basics

$\mathbb{N}$	Set of natural numbers.
$\mathbb{R}$	Set of real numbers.
$\wedge$	The logical conjunction.
$\vee$	The logical, inclusive disjunction.
$\neg$	The logical negation.

Relational Algebra

$\mathcal{R}$	A relation. Sometimes with subscript, e.g., $\mathcal{R}_{\texttt{name}}$ to specify name or index.
$\mathcal{R}^{\mathcal{O}}$	A ranked, relation that exhibits an ordering of elements induced my $\mathcal{O}$. Sometimes used with subscript, e.g., $\mathcal{R}_{\texttt{name}}$ to specify name or index.
$\mathcal{D}$	A data domain of a relation. Sometimes with subscript, e.g., $\mathcal{D}_{\texttt{name}}$, to specify name or index.
$\mathcal{A}$	An attribute of a relation, i.e., a tuple $(\texttt{name}, \mathcal{D})$. Sometimes with subscript to specify name or index.
$\mathcal{A}^*$	An attribute that acts as a primary key.
$\overline{\mathcal{A}}$	An attribute that acts as a foreign key.
t	A tuple $t \in \mathcal{R}$ of attribute values $a_i \in \mathcal{D}_i$. Sometimes with subscript to specify the index of the tuple in $\mathcal{R}$.
$\mathbb{D}$	The set system of all data domains $\mathcal{D}$ supported by a DBMS.
$\texttt{SCH}$	The schema of a relation $\mathcal{R}$, i.e., all attributes $\texttt{SCH}(\mathcal{R}) = (\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_N)$ that make up $\mathcal{R}$.
σ	The unary selection operator used to filter a relation $\mathcal{R}$. Usually printed with a subscript to describe the boolean predicate, e.g. $\sigma_{\mathcal{P}}$.
π	The unary projection operator used to restrict the attributes $\mathcal{A}$ of a relation $\mathcal{R}$. Usually used with subscript to list the desired attributes , e.g. $\pi_{\mathcal{A}_1, \mathcal{A}_2}$.

Multimedia Retrieval

$\hat{\delta}$	A Distance Function Class (DFC) $\hat{\delta}$: $\mathcal{D}_q \times \mathcal{D}_q \times \mathcal{D}_1 \cdot \times \mathcal{D}_n \rightarrow \mathbb{N}$

Part I

Introduction

1

Introduction

The term *multimedia* describes the combination of different forms of digital media – also called *modalities* – into a single, sensory experience that carries a higher level semantic. Those modalities include but are not limited to images and videos (visual), music, sound effects and speech (aural) or textual information. However, more exotic media types such as 3D models or signals produced by sensors can also be seen as modalities, even though experience by a human consumer may depend on pre-processing by specialized hard- and software.

Nowadays, people encounter digital media and multimedia on a daily basis when watching videos on Netflix or YouTube, when listening to music on Spotify or when browsing a private image collection on their laptop. (Multi-)media content makes up a large part of today's Internet and constitutes a major driving force behind its growth, as both volume and variety increases at an ever increasing pace. An important contributing factor are social media platforms, where users act both as consumers and producers of digital content. Current estimates suggest, that there are roughly 4.66 billion active Internet users worldwide, of which 4.2 billion can be considered active social media users[1]. Facebook alone contributed to 144 thousand uploaded images per minute in 2020. And many more of these platforms, such as Instagram or Twitter, serve millions of users with mixed, self-made content involving text, images, videos or a combination thereof. A similar study found, that by 2025 we will produce a yearly amount of 175 Zettabytes (i.e, 10^{21} bytes) worth of data[2].

Looking at these numbers, the need for efficient and effective tools for *managing*, *manipulating*, *searching*, *exploring* and *analysing* multimedia data corpora becomes very apparent, which has given rise to different areas of research.

[1] Source: Statista.com, "Social media usage worldwide", January 2021
[2] Source: Statista.com, "Big Data", January 2021

1.1 Working with Multimedia Data

On a very high level, multimedia data collections consist of individual multimedia items, such as video, image or audio files. Each item, in turn, comprises of *content*, *annotations* and *metdata*. Unlike traditional data collections that contain only text and numbers, the content of the multimedia item itself is unstructured on a data level, which is why *feature representations* that reflect a media item's content in some way and that can be handled by data processing systems are required [ZW14]. Traditionally, such feature representations have often been numerical vectors $f_i \in \mathbb{R}^d$. However, in theory, any mathematical object that can be processed by a computer can act as a feature.

Multimedia analysis, which has its roots in *computer vision* and *pattern recognition* and started in the early 1960s and deal with the automated, computer-aided analysis of visual information found in images and later videos, i.e., the extraction and processing of feature representations. In the early days of computer vision, a lot of effort went into the engineering of feature representations that captured certain aspects of a media item's content, such as the colour distribution, texture or relevant keypoints [Low99; BTVG06] in an image. Once such features have been obtained, they can be used to perform various tasks such as classification, clustering or statistical analysis. With the advent of deep learning, the extraction of features could largely be automated through neural network architectures such as the *Convolutional Neural Network (CNN)* and sometimes even be integrated with the downstream analysis [GBC16].

Obviously, such analysis is not restricted to the visual domain and can be applied to other types of media such as speech, music, video or 3D models with specific applications, such as, speech recognition, audio fingerprinting in music, movement detection in videos or classification of 3D models, all of which fall into the broader category of multimedia analysis.

1.1.1 Multimedia Retrieval

Traditionally, multimedia retrieval or content-based retrieval could be seen as a special niche within the multimedia analysis domain. It constitutes a dedidcated field of research that deals with searching and finding items of interest within a large (multi-)media collection. Even though this may sound like the main function of a database, it is a very different task for multimedia than it is for structured data [BdB+07]. On the one hand, given the structure of a relational database and languages like SQL, a user can specify exactly what ele-

ments from the database should be selected using predicates that either match or don't match the items in a collection. For example, when considering a product database that contains price information for individual items, it is trivial to formulate a query that selects all items above a specific price threshold.

Retrieving multimedia data, on the other hand, comes with indirections due to the unstructured nature of the content, the feature representations used as a proxy for it and the *semantic gap* associated with these representations. A very popular model to work with the feature representations involves calculation of (dis-)similarity scores from the features and sorting and ranking of items based on this score. This is commonly referred to as the *vector space model* of multimedia retrieval and similiarty search. Over the years, many different combinations of features and ranking models have been proposed to facilitate content-based retrieval of different media types, such as, images, audio or video as have been different types of query formulation, such as *Query-by-Sketch*, *Query-by-Humming* or *Query-by-Sculpting* [CWW+10; GLC+95; BGS+20].

Add sources: Survey for each modality

Furthermore, when looking at concrete system implementations that facilitate interactive multimedia retrieval for an end-user today, the lines between multimedia retrieval and multimedia analytics quickly start to blur. This is because, in addition to the extraction of appropriate features and the conception of effective ranking algorithms, multimedia retrieval systems today also concern themselves with aspects such as query (re-)formulation and refinement, results presentation and efficient exploration [LKM+19]. In addition, multimedia retrieval systems do not simply operate on features representations anymore but combine *similarity search* on features and *Boolean retrieval* on annotations and metadata [RGL+20]. Therefore, one could argue that multimedia retrieval systems perform a very specific type of multimedia analytics task, which is that of finding unknown items that satisfy a specific information need. This makes all the arguments made about data processing and data management requirements for multimedia analytics applicable to multimedia retrieval as well.

1.1.2 Multimedia Analytics

Multimedia analytics aims at generating new knowledge and insights from multimedia data by combining techniques from multimedia analysis and visual analytics. While multimedia analysis deals with the different media types and how meaningful representations and models can be extracted from them, visual analytics deals with the user's interaction with the data and the models themselves [CTW+10; KKE+10]. Simply put, multimedia analytics can be seens as a

back and forth between multimedia (data) analysis and visual analytics, wheras analysis is used to generate models as well as visualisations from data which are then examined and refined by the user and their input. This is an iterative process that generates new knowledge and may in and by itself lead to new information being attached to the multimedia items in the collection.

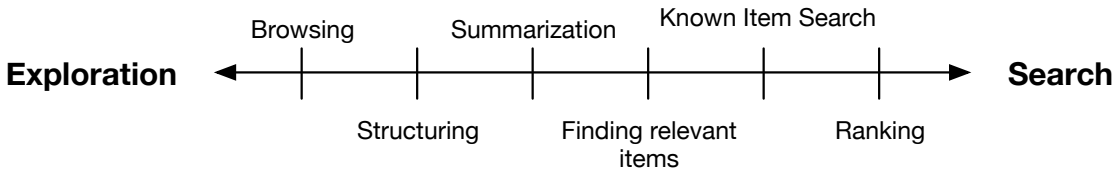

Figure 1.1 Exploration-search axis of multimedia analytics [ZW14].

For analytics on a multimedia collection, Zahalka et al. [ZW14] propose the formal model of an *exploration-search axis*, which is depicted in Figure 1.1. The model is used to characterize the different types of tasks carried out by the user. The axis specifies two ends of a spectrum, with *exploration* marking one end – in case the user knows nothing about the data collection – and *search* marking the other end – in case the user knows exactly which specific items of a collection they're interested in. During multimedia analytics, a user's activities oscillate between the two ends of the spectrum until the desired knowledge has been generated. Unsurprisingly, all of the depicted activites come with distinct requirements on data transformation and processing.

As data collections become large enough for the relevant units of information – i.e., feature representations, annotations and metadata – to no longer fit into main memory, multimedia analytics and the associated data processing quickly becomes an issue of scalable data management [JWZ+16]. This data management aspect becomes very challenging when considering the volume and variety of the multimedia data, the velocity at which new data is generated and the inherently unstructured nature of the media data itself.

1.1.3 Dynamic Data Management

Finetune!

It is important to emphasize, that a media item can comprise of all of the aforementioned components and that a multimedia collection may contain items of different types. Furthermore, when looking at a media item's lifecycle, all of the aforementioned aspects are not static; annotations, metdata and features may either be generated upon the item's creation (e.g., for technical metadata), as a

result of data-processing and analysis or by manually adding the information at some stage. Hence, any data management system must be able to cope with changes to that information. Those requirements are formalized in Section 3.1.

1.2 Research Gap and Objective

It has been pointed out by Jonson et al. [JWZ+16] (p. 296) that "Multimedia analytics state of the art [...] has up to now [...] not explicitly considered the issue of data management, despite aiming for large-scale analytics.". Despite recent advances and the development of concrete architecture models for multimedia database management [GS16; Gia18] and multimedia retrieval systems that refactor data management into distinct components [Ros18], that statement, to some extent, still holds true today. While [Gia18] makes important contributions towards a unified data-, query- and execution model required for effective search and exploration in multimedia collections, scalability aspects and the need for near real-time query performance, especially in the face of dynamic data, are not systematically considered. On the contrary, the proposed models – despite being seminal for data management in certain multimedia retrieval applications – postulate assumptions, that have considerable impact on the practical applicability of data management systems implementing them.

The starting point for the research described in this thesis is therefore the current state-of-the-art for data management in multimedia retrieval and analytics as briefly touched upon in the previous sections. Starting from and inspired by the models and solutions proposed in [GS16; Gia18] and motivated by the "Ten Research Questions for Scalable Multimedia Analytics" [JWZ+16], this thesis challenges three basic assumptions currently employed and operated upon in multimedia data management and explores the ramifications of doing so, with the higher level goal of bridging certain gaps between research conducted in multimedia retrieval, analysis and analytics on the one hand, and classical data management and databases on the other. These assumptions are namely:

Assumption 1: Staticity of data collections Most multimedia retrieval systems today make a distinction between an *offline* phase during which media items are analysed, features are generated and derived data is ingested into a data management system, and an *online* phase, during which queries of the data management system take place. Usually, no changes to the data collection are being made during the online phase. This model is proposed by both [Gia18]

and [Ros18] and to the best of our knowledge, most existing multimedia retrieval and analytics systems implement this either explicitly or implicitly. This simplification allows for time consuming processes related to feature extraction and indexing to take place separated from any concurrent query activities and eases requirements on transaction isolation.

Assumption 2: Nearest neighbor search The vector space model used in multimedia retrieval relies on a notion of similarity search that is usually expressed as finding the k nearest neighboring feature vectors $\vec{v}_{i\in[1,k]} \in \mathrm{C}$ to a query vector $\vec{q} \in \mathbb{R}^d$ in a collection $\mathrm{C} \subset \mathbb{R}^d$ given a certain distance function. Very often, metrics such as the Euclidean or the Manhattan distance are employed for this comparison. While this model is very concise, computationally efficient and rather simple, it merely allows for the ranking of potential results and, given that the underlying model and the query is precise enough, finding the relevant or desired item(s).

Assumption 3: User defines execution Database management systems usually evaluate and select the execution plan for an incoming query during a step that is refered to as *query planning*. The underlying assumption here is that the database system has all the information required to determine the most effective execution path in terms of cost parameters such as required I/O, CPU and memory usage. In multimedia retrieval, this is not the case since, for example, index selection relies on a lot of different aspects that, to some extent, can be parametrized by the client issuing a query or that may be subject to change. Therefore, the index used for executing a query is often selected explicitly by the user issuing the query.

It is worth noting, that Assumption 1 and 3 both go against well-established design principles usually found it modern database systems [Pet19]. While it may be convenient from a perspective of system design, to assume a data collection to be static, such a mode of operation is utterly limiting when considering data that is subject to change, as is the case, for example, when doing analytics or when having an application with CRUD support. A similar argument can be made for manual index selection. Such an assumption may be simplifying the process of query planning but assumes, that a user is always a technical expert. Furthermore, it limits the amount of optimization that can be applied by the data management system especially in the face of non-static data collections, where indexes are changing, or changing query workloads.

As for Assumption 2, one can state that the described model is only able to accommodate the search-end of the *exploration-search axis*, assuming that features are, in fact, real valued vectors. It quickly becomes unusable for tasks such as browsing, structuring and summarization, delegating the required data processing to upper-tier system components. Refering to [JWZ+16], it would however be desirable to offer such primitives at the level of the data management system.

Transition from assumptions via overarching research goal to research question.

1.2.1 Research Questions

Challenging the aforementioned assumptions raises very specific questions that fundamentaly impact the design of a *multimedia data management system*. These questions are briefly summarized in Table 1.1.

Table 1.1 List of research questions (RQ) resulting from challenging assumptions(AS) one, two and three.

RQ	Question	Assumption	Domain
1	Which commonly used, secondary index structures for NNS (e.g., VA [WSB98], LSH [IM98], PQ [JDS11] based indexes) can cope with changes to data and to what extent?	AS 1	
2	Can we estimate and quantify deterioration of retrieval quality of index structures from RQ1 as changes are being made to the underlying data collections?	AS 1	
3	How can we handle index structures from RQ1 for which to expect deterioration during query planning and execution?	AS 1	
4	Can we devise a model that (temporarily) compensates deterioration of retrieval quality of index structures?	AS 1	
5	How can user knowledge about the the retrieval task at hand be factored into query planning without forcing the user the make explicit choices about how a query should be executed?	AS 1 & 3	
6	How would a cost model that factors in desired retrieval accuracy look like and can it be applied during query planning?	AS 3	
7	Assuming the cost model in RQ6 exists, at what levels of the system can it be applied (globally, per query, context)?	AS 3	
8	Is there a measurable impact (e.g., on query execution time vs. accuracy) of having such a cost model?	AS 3	
9	Can we generalize the model for similarity search (i.e., the vector space model) and what is the consequence of doing so?	AS 2	
10	Do the existing applications and use-cases justify a generalization?	AS 2	

Associated research questions to domains (retrieval, analytics, dynamic).

RQ1 to RQ5 address the issue of index structures for NNS, which are mostly unable to cope with data that is subject to change, since their correctness deterio-

rates as data is modified. The focus of these questions are whether deterioration can be quantified and how it can be handled by a system. We argue, that both is necessary for practical application in dynamic data management.

RQ6 to RQ8 explore the possibility of a cost model, that takes accuracy of the produced results into account. Since most techniques for fast NNS rely on approximation, inaccuracy is an inherent factor for such operations. Assuming such a model exists, it can be used by a user or system administrator to make explicit choices between either accuracy or execution peformance. In addition, such a cost model can be put to use when deciding what indexes to use in face of deteriorated retrieval quality due to changing data.

And finally, RQ9 and RQ10 address the issue of a more generalized model for similarity search and the impact of such a model on all the different system components. Most importantly, however, they explore and justify the need for such a model, which is not self-evident, based on concrete use-cases and applications.

1.3 Contribution

In this thesis, we try to address the research gap identified and described in Section 1.2 and thereby try to bridge the disparity between the fields of databases and retrieval systems. The contribution of this thesis can be summarized as follows:

- We examine the impact of challenging the *data staticity assumption* on index structures commonly used for nearest neighbor search. Most importantly, we describe a model to *quantify* the effect of changing data for commonly used structures and to *expose* that information to the data management system.
- We describe an *adaptive index management* model by which a data management system can compensate errors introduced at an index level due to changes to the underlying data. The main design goal for that mechanism is, that it can be employed regardless of what type of index is used underneath.
- We propose a *cost-model that factors-in accuracy* of generated results in addition to common performance metrics, such as IO-, CPU-, or memory-usage and, based on that model, derive mechanisms for the user to express their preference for either accuacy or speed at different levels of the system.

- We postulate a more *generalized model for similarity search* and explore implications of such a model on aspects, such as, query planning.
- We introduce a working implementation that implements the aforementioned models, in the form of *Cottontail DB* [GRH+20].
- We present an evaluation of the impact of the model changes on real world datasets to provide a basis upon which their applicability can be assessed.

The described contributions are presented in four parts: The first part, to which this introduction belongs, introduces the problem and provides motivating use-cases and applications (Chapter 2) as well as an overview of relevant research done in the fields of multimedia retrieval, multimedia analytics and data management (Chapter 8).

The second part, gives a brief summary of the theoretical foundation in multimedia analysis & retrieval (Chapter 3) and databases (Chapter 4) required to understand the remainder of this thesis. The function of these chapters is that of a refresher for readers not familiar with either domain.

The third part introduces the theoretical models that make up the aforementioned contributions (Chapter 5) and introduces our reference implementation Cottontail DB (Chapter 6).

The fourth and final part presents the evaluation using Cottontail DB as an implementation (Chapter 7) and discusses the conlusions and potential future work (Chapter 9)

2

Applications and Use Cases

2.1 Use case 1: Multimedia Retrieval System

vitrivr, vitrivr VR with focus on search and exploration and data mangement & query implications (e.g., for SOMs, staged querying etc.). It remains to be seen how changes to data can be motivated here.

2.2 Use case 2: Analysis of Social Media Streams

Online analysis in Pythia, Delphi. Mainly as a motivating use case for why data may be subject to change.

Demo paper!

2.3 Use case 3: Magnetic Resonance Fingerprinting (MRF)

MRF as a concrete example why the classical NNS is too limited for certain use cases and an extension should be considered.

Paper!

Part II

Foundations

3

On Multimedia Analysis and Retrieval

3.1 Multimedia Data and Multimedia Collections

Formalisation of what multimedia data is and what forms it can take (video, audio, images, text + metadata etc.). This formal model has the potential of being an original contribution, since we will make very explicit assumptions about what aspects of a multimedia item there are and which ones are mutable or immutable (e.g, content vs. annotations, metadata, features etc.)

3.2 Multimedia Retrieval

3.2.1 Similarity Search and the Vector Space Model

In multimedia retrieval, there are two important assumptions for similiarity search. These assumptions can be summarized as follows:

- For every object c_i in a (multimedia) collection C, there exists a feature transformation $\phi : C \rightarrow \mathcal{F}$, that maps the object $o_i \in C$ to a feature space $\mathcal{F}$.
- The feature space $\mathcal{F}$ and a to be defined distance function $\delta : \mathcal{F} \times \mathcal{F} \rightarrow \mathbb{R}_{\geq 0}$, constitute a metric space $(\mathcal{F}, \delta)$, thus satisfying the non-negativity, identity of indiscernibles, symmetry and subadditivity condition.

The output of δ – i.e., the calculated distance d – acts as a proxy for (dis-)similiarty between two objects o_i, $o_j \in C$ given the feature transformation ϕ. Hence, the closer two objects o_i, o_j appear under the transformation, the more

(dis-)similar they are. For the sake of completeness, it must be pointed out, that whether similarity is directly or inversely proportional to the distance is a matter of definition and depends on the concrete application. Also, in practice, multiple feature transformations ϕ_m may exist for a given media collection, leading to different feature spaces $\mathcal{F}_m$ for a collection C that must be considered jointly. Both aspects are usually addressed by additional *correspondence* and *scoring* functions.

Correspondence and score functions, score fusion

3.2.2 Approximate Nearest Neighbor Search

Describe techniques for approximate nearest neighbor search (ANN). Focus on a more conceptual overview of the types of algorithms rather than just enumerating concrete examples; this can be used as a build-up for discussing properties of different index structures later.

3.2.3 Beyond Similarity Search

Retrieval and analytics techniques that go beyond simple similarity search (e.g. SOM, summarization, clustering)

3.3 Online Multimedia Analysis

Introducing an online analysis pipeline (e.g., Pythia / Delphi).

3.4 Multimedia Analytics

Describe how the combination of analysis

3.4.1 Beyond Similarity Search

Under third normal form, a non-key field must provide a fact about the key, use the whole key, and nothing but the key.

— William Kent

4

On The Design of a Database Management System

Database Management System (DBMS), or simply "databases", power everything from small and simple websites to large data warehouses that serve millions of users in parallel. Database systems play a crucial role in banking, e-commerce, science, entertainment and practically every aspect of our socio-economic lives. The first commercial Database Management System (DBMS) were introduced in the 1960s [GUW09] and they have evolved ever since to adapt to a wide range of requirements. Even though many different flavours of DBMS have emerged over the years, at their core, they still serve the same, fundamental purpose:

Management DBMS enable their users to organize data corpora that can range from a few megabytes to houndreds of terrabytes according to a defined data model. For example, data can be structured into documents, tables, trees or graphs that in turn can be organized into collections or schemata.

Definition DBMS provide users with the ability to alter the organization of their data within the constraints of the data model using a Data Definition Language (DDL).

Manipulation DBMS provide users with the ability to modify the data within the constraints of the data model using a Data Manipulation Language (DML). Modifications may include adding, removing or changing entries.

Querying DBMS provide users with the ability to query the data using a Data Query Language (DQL). Such queries can be used to answer specific "questions" about the data.

Guarantees DBMS usually provide guarantees, such as assuring durability upon failure or providing access control for concurrent read and write operations. A well-known set of guarantees offered by many modern DBMS is known by its acronym ACID – which stands for **A**tomicity, **C**onsistency, **I**solation and **D**urability [HR83].

Since the contributions of this Thesis rely on decades of database research, we use this chapter to provide a brief overview over the relevant fundamentals and related work. Most of the fundamental aspects are inspired and guided by [GUW09] and [Pet19].

4.1 Data Management and Data Models

A data model is a formal framework that describes any type of data or information. It usually involves a formal description of the data's *structure*, the *operations* that can be performed and the *constraints* that should be imposed on the data [GUW09]. The purpose of any data model is to formalize how data governed by it can be accessed, modified and queried and any given DBMS usually adopts a specific type of data model.

In the context of database systems and data management, it has become common practice to distinguish between different levels of abstraction for data models, as can be seen in Figure 4.1. At the top, there is the *conceptual data model*, which is often closely tied to some real-world scenario. For example, in an online shop, one can think in terms of customers that order things, products that are being sold and orders that have been placed. In the Entity Relationship Model (ERM) [Che76] – a popular framework used to describe conceptual data models – those "real things" would be modeled as *entities* that come with dedicated *attributes* that describe them and can have certain *relationships* among them as required by the concrete application.

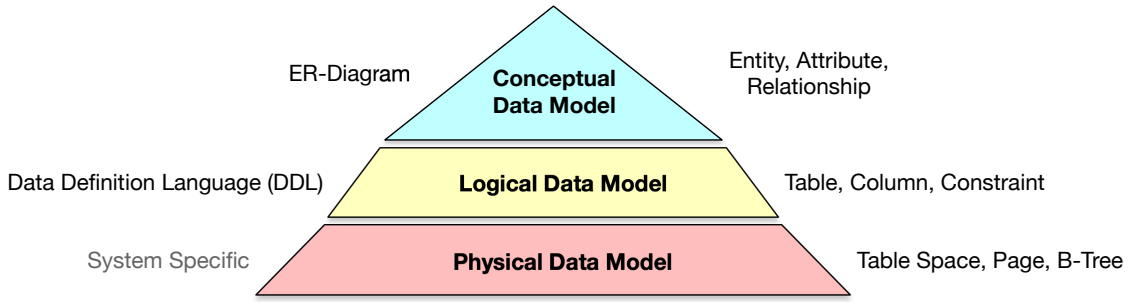

Figure 4.1 Different levels of abstraction for data modelling.

The *logical data model* is more closely tied to the type of database being used. For example, one could store each entity described before in a dedicated table wherein each attribute occupies a different column. At this level, one can use DDL to describe and modifiy the data organization and DQL or DML to query and modify the data. A very important example language that covers all three aspects would be the Structured Query Language (SQL), which became an international standard in 1987 under *ISO 9075* and has evolved ever since [Cha12].

At the very bottom, we usually find the *physical data model*, which is specific to the database implementation and describes low-level aspects such data access in terms of *read* and *write* operations to data structures such as *table spaces*, *data pages* or *trees*. An important argument in favour of separating logical from physical data model, even though they are both somewhat specific to a DBMS implementation, is that the end-users should not concern themselves with how exactly data is organized and accessed at the lowest level but should instead describe their intent in terms of the information they are trying to access. Mapping this user-intent to low-level data access operations is then a task of the DBMS.

4.2 The Relational Data Model

In June 1970, E. F. Codd published his pivotal research article *Relational Model of Data for Large Shared Data Banks* [Cod70] in which he describes a logical data model, which he himself refers to as "relational". This model has become the fundament on which many modern database management systems have been built in the past decades, with early examples dating back to the 1970s [ABC+76] and promintent, contemporary examples including systems such as *Maria DB*[1], *PostgreSQL*[2], *Oracle* or *MS SQL*.

The relational model is structured around *relations*, which are a mathematical construct but can be visualized as two-dimensional tables. Such a table consists of columns – which are called *attributes* – and rows – which are called *tuples* and hold *attribute values*. Semantically, a relation can be seen as a knowledge-base about some fact – such as, the paintings held by a museum – under a Closed-world assumption (CWA). That is, the relation contains all the information available about the fact and can thus be used do derive conclusive answers or results given a stated question or query.

Reference

[1] Open Source, see https://mariadb.org/
[2] Open Source, see https://www.postgresql.org/

In order to formalize the structure of and the operations that can be executed on the data represented by a relation, one can use Definition 4.1.

Definition 4.1 Relation according to [Cod70]

Let $\mathcal{D}_j$ be sets we call *data domains* with $j \in [1, N]$ and $N \in \mathbb{N}_{>0}$. A *relation* $\mathcal{R}$ constructed over these data domains is a set of M *tuples* $t_i = (a_{i,1}, a_{i,2}...a_{i,N})$ with $i \in [1, M]$ and $M \in \mathbb{N}_{>0}$ such that the first attribute value of any tuple comes from $\mathcal{D}_1$, the second from $\mathcal{D}_2$ and so forth. That is, values $a_{i,1} \in \mathcal{D}_1, a_{i,2} \in \mathcal{D}_2...a_{i,N} \in \mathcal{D}_N$. Ergo, a relation can be seen as a subset of the cartesian product of all data domains over which it was constructed, i.e., $\mathcal{R} \subset \mathcal{D}_1 \times \mathcal{D}_2... \times \mathcal{D}_N$.

The number of data domains N is referred to as its *degree* and we call such a relation *N-ary*. The number of tuples M is called the *cardinality* of $\mathcal{R}$ with $M = |\mathcal{R}|$. It must be noted, that the relational model does not dictate what the data domains $\mathcal{D}$ are. However, in a relational DBMS, the they usually correspond to the data types supported by the database and the programming environment it was written in, for example, integer and floating point numbers or text. In this Thesis, we therefore sometimes use *data domain* synonymous for *data type* [3].

In addition to Definition 4.1, we will use the identities 4.2 to 4.5 throughout this Thesis when working with relations.

Definition 4.2 Attributes of a Relation

Let $\mathcal{R}$ be a N-ary relation over the data domains $\mathcal{D}_i$, $i \in [1, N]$ with corresponding names or indexes l_i. We call the combination of a data domain with a human readable label *attribute*, that is, $\mathcal{A}_i = (\mathcal{D}_i, l_i)$. To simplify notation, we sometimes use the label of an attribute in the subscript instead of the index.

Definition 4.3 Schema of a Relation

Let $\mathcal{R}$ be a N-ary relation over the attributes $\mathcal{A}_i$, $i \in [1, N]$. We call the tuple of all attributes over which $\mathcal{R}$ was constructed the *heading* or *schema* $\texttt{SCH}(\mathcal{R})$ of $\mathcal{R}$.

$$\texttt{SCH}(\mathcal{R}) = (\mathcal{A} : \text{if } \mathcal{A} \text{ is an attribute of } \mathcal{R})$$

[3] Strictly speaking, the data domains of a given relation $\mathcal{R}$ are merely subsets of the sets that represent the respective data type which, in turn, are subsets of even more basic sets such as $\mathbb{N}$ or $\mathbb{R}$ for `Int` and `Float` respectively. The relationship between types and data domains is subject of a more categorical approach to data models and nicely layed out in [Spi09]

Definition 4.4 Accessing Attribute Values

Let $\mathcal{R}$ be a N-ary relation over attributes $\mathcal{A}_i$, $i \in [1, N]$ and let further $t \in \mathcal{R}$. To address the attribute value $a_i \in t$ that belongs to attribute $\mathcal{A}_i \in \texttt{SCH}(\mathcal{R})$, we use the *attribute value accessor* $t\,[\mathcal{A}_i]$.

$$a_i = t\,[\mathcal{A}_i] = \{a_j \in t : i = j\}$$

Definition 4.5 Supported Data Domains

For a given DBMS, we call $\mathbb{D}$ the set system of data domains or data types supported by the system.

$$\mathbb{D} = \{\mathcal{D} : \text{if } \mathcal{D} \text{ is supported by DBMS}\}$$

In its original form, the relational model assumes the following properties to be true for a relation $\mathcal{R}$ and its attributes [Cod70]:

Ordering of Tuples Tuples $t \in \mathcal{R}$ are inherently unordered and two relations are considered equal if they contain the same tuples, regardless of order.

Ordering of Attributes Attribute values a_i always occur in the same order within the tuple $t \in \mathcal{R}$, which corresponds to the order of the attributes $\mathcal{A}_i \in \texttt{SCH}(\mathcal{R})$. This order can evolve over time but remains constant in a momentary snapshot of $\mathcal{R}$. It follows from the definition that $|t| = |\texttt{SCH}(\mathcal{R})| \forall t \in \mathcal{R}$

Duplicates Since relations are sets, they do not allow for duplicates, i.e., every tuple $t \in \mathcal{R}$ must be unique in terms of their attribute values.

Given the idea of a relation, the sum of all data managed by a DBMS can logically be regarded as a collection of different relations $\mathcal{R}_k$, $k \in \mathbb{N}_{\geq 0}$ of assorted degrees N_k over data domains $\mathcal{D} \in \mathbb{D}_{\texttt{dbms}}$ (i.e., a collection of tables). The schema of a database can then be seen as the set system of all $\texttt{SCH}(\mathcal{R}_k)$. As [Cod70] points out, relations are subject to change over time. These changes can take place on the level of any relation's structure, i.e., $\texttt{SCH}(\mathcal{R}_k)$, the relations $\mathcal{R}_k$ itself, e.g., by tuples being added to (insert) or removed from (delete) a relation or the level of a tuple $t \in \mathcal{R}_k$, e.g., by altering one or multiple attribute values.

Example 4.1 features an example relation `paintings` visualized as a table. Each entry in the table represents a painting and the related attribute values.

Example 4.1 Table Representation of a Relation $\mathcal{R}_{\texttt{paintings}}$

The following table lists the content of a ternary relation ($N = 3$) $\mathcal{R}_{\texttt{paintings}}$. The attributes $\mathcal{A}_{\texttt{title}}, \mathcal{A}_{\texttt{artist}}, \mathcal{A}_{\texttt{painted}}$ correspond to the table's columns. The individual tuples t_i are "valid" combinations of painting title, artist and year of conception under CWA and constitute the rows.

$\mathcal{R}_{\texttt{paintings}}$	$\mathcal{A}_{\texttt{title}}$	$\mathcal{A}_{\texttt{artist}}$	$\mathcal{A}_{\texttt{painted}}$
t_1	Mona Lisa	Leonardo da Vinci	1506
t_2	The Starry Night	Vincent van Gogh	1889
t_3	Las Meninas	Diego Velázquez	1665

4.2.1 Keys and Normal Forms

The notion of a relation provides us with the basic tools for data modeling. In addition to Definition 4.1, Codd proposed a range of constraints to guarantee proper data definition using the relational model and the following constructs [Cod70]:

A Primary Key (PK) $\mathcal{P}$ of relation $\mathcal{R}$ is a subset of attributes $\mathcal{P} \subset \texttt{SCH}(\mathcal{R})$ that uniquely identify a tuple $t \in \mathcal{R}$. That is, the attributes that are not part of the PK and the associated values are functionally determined by $\mathcal{P}$. Using Example 4.2, the painting and all its attributes, i.e., the artist and the year of its creation, are functionally determined by the name of the painting, which is assumed to uniquely identify the entry.

A Foreign Key (FK) $\mathcal{F}$ of relation $\mathcal{R}$ is a subset of attributes $\mathcal{F} \subset \texttt{SCH}(\mathcal{R})$ that are not member of a PK, i.e., $\mathcal{F} \cap \mathcal{P} = \emptyset$, but reference the PK of $\mathcal{R}$ or some other relation R^*. FK can be used to model relationships between relations. Using Example 4.2, the artist that created a painting is referenced through a FK $\overline{\mathcal{A}}_{\texttt{artist}}$ in $\mathcal{R}_{\texttt{painting}}$ that references the PK ${\mathcal{A}^*}_{\texttt{artist}}$ in $\mathcal{R}_{\texttt{artist}}$.

An example of relations with primary and foreign key attributes is given in Example 4.2. PKs and FKs are indicated with star and overline respectively. Note, that we assume here that people (artists) and paintings are uniquely identified by their first and lastname and their title respectively, which is obviously a simplification. In reald-world applications, often artificial PKs are being generated to avoid unintended collisions.

Example 4.2 Relations with Primary and Foreign Keys

The following tables lists the schema and extent of $\mathcal{R}_{\texttt{paintings}}$ and $\mathcal{R}_{\texttt{artists}}$.

$\mathcal{R}_{\texttt{painting}}$	$\mathcal{A}^*{}_{\texttt{title}}$	$\overline{\mathcal{A}}_{\texttt{artist}}$	$\mathcal{A}_{\texttt{painted}}$
t_1	Mona Lisa	Leonardo da Vinci	1506
t_2	The Starry Night	Vincent van Gogh	1889
t_3	Las Meninas	Diego Velázquez	1665

$\mathcal{R}_{\texttt{artist}}$	$\mathcal{A}^*{}_{\texttt{artist}}$	$\mathcal{A}_{\texttt{birth}}$	$\mathcal{A}_{\texttt{death}}$
t_1	Leonardo da Vinci	1452	1519
t_2	Vincent van Gogh	1853	1890
t_3	Diego Velázquez	1599	1660

Given the notion of primary and foreign keys, Codd proposed a series of *normal forms* that are an indication of the quality of a data model based on the functional dependency of the attributes on a PK. The basic idea is to avoid redundancy by normalization, i.e., by putting data that belongs together into dedicated relations and modeling relationships between them using foreign keys. The first three normal forms are as follows:

First Normal Form (1NF) requires, that no attribute in a relation has other relations as values, i.e., attributes are atomic and instead, relationships can be established by means of FKs. Given $\mathcal{R}_{\texttt{painting}}$ in Example 4.2, this means that $\mathcal{R}_{\texttt{artist}}$ cannot be stored as an attribute of $\mathcal{R}_{\texttt{painting}}$ (which would be possible according to Definition 4.1, since data domains can hold any type of element) and instead requires a dedicated relation.

Second Normal Form (2NF) requires, that every non-prime attribute is functionally determined by the whole primary key and not any subset thereof. Given $\mathcal{R}_{\texttt{artist}}$ in Example 4.2 and assuming that $\mathcal{A}^*{}_{\texttt{artist}}$ was split into two attributes $\mathcal{A}^*{}_{\texttt{firstname}}$ and $\mathcal{A}^*{}_{\texttt{lastname}}$ (composite key), this means that all of the non-prime attributes must be determined by $\mathcal{A}^*{}_{\texttt{firstname}}$ and $\mathcal{A}^*{}_{\texttt{lastname}}$ jointly, i.e., the full name, and not just either $\mathcal{A}^*{}_{\texttt{firstname}}$ or $\mathcal{A}^*{}_{\texttt{lastname}}$.

Third Normal Form (3NF) requires, that every non-prime attribute is functionally determined solely by the primary key and that they do not depend on any other attribute. Given $\mathcal{R}_{\texttt{artist}}$ in Example 4.2, this means that all of the non-prime attributes must be determined by $\mathcal{A}^*{}_{\texttt{artist}}$ alone.

All the normal forms build onto one another, i.e., for a data model to be considered 3NF is also required to satisfy 2NF and 1NF[4]. Additional normal forms up to *6NF* and the Boyce–Codd Normal Form (BCNF), a slightly stronger version of 3NF, have been defined. For the sake of brevity, we will ommit those since they are not relevant for the discussion ahead.

4.2.2 Relational Algebra

Having introduced the aspect of data representation and constraints, we can now move to that of operations that can be performed on the data upon querying. For that purpose, [Cod70] proposed the idea of a *relational algebra*, which follows a simple yet powerful idea: All query operations performed on relations are expressed by a set of *relational operators* that take one or multiple relations as input and output a new relation as expressed by Equation (4.1).

$$\texttt{OP} : \mathcal{R}_1, \ldots, \mathcal{R}_n \rightarrow \mathcal{R}_O \tag{4.1}$$

Those relational operators can then be composed to express a query of arbitrary complexity as indicated by Equation (4.2).

$$\texttt{QUERY} = \texttt{OP}_1 \circ \texttt{OP}_2, \ldots, \circ \texttt{OP}_\texttt{m} \tag{4.2}$$

In addition to this idea, Codd proposed a minimal set of relational operators listed in Table 4.1 and explained in the following sections. For the sake of completeness, it must be mentioned that notation and operators may slightly differ depending on the literature. We mainly use [GUW09] as our reference.

4.2.2.1 Simple Set Operations

Since relations are in essence sets of tuples, all basic operations known from set theory can be applied, namely *union*, *intersection* and *difference*, with the only constraint that the two input relations $\mathcal{R}_L, \mathcal{R}_R$ must be *union compatible* and thus exhibit the same attributes, i.e., $\texttt{SCH}(\mathcal{R}_L) = \texttt{SCH}(\mathcal{R}_R)$.

The set union $\mathcal{R}_L \cup \mathcal{R}_R$ generates a new relation of all tuples contained in either $\mathcal{R}_L$ OR $\mathcal{R}_R$, as expressed by Equation (4.3)

$$\mathcal{R}_L \cup \mathcal{R}_R = \{t : t \in \mathcal{R}_L \vee t \in \mathcal{R}_R\} \tag{4.3}$$

[4] This has led to the mnemonic *"The key, the whole key, and nothing but the key, So help me Codd."* in reference to a similarly structured oath often used in courts of law.

Table 4.1 The relational operators proposed by Codd et al [Cod70; GUW09].

Name	Symbol	Arity	Example
Union	$\cup$	2	Set union of two input relations.
Intersection	$\cap$	2	Set intersection of two input relations.
Difference	$\setminus$	2	Set difference of two input relations.
Cartesian Product	$\times$	2	Pairs each tuple from one the left with every tuple of the right input relation and concatenates them.
Rename	$\rho_{\mathcal{A}_A \to \mathcal{A}_B}$	1	Renames attribute $\mathcal{A}_A$ in input relation to $\mathcal{A}_B$.
Selection	$\sigma_{\mathcal{P}}$	1	Removes tuples from the input relation that don't match predicate $\mathcal{P}$.
Projection	π_{χ}	1	Removes attributes from the input relation that are not included in χ.
Natural join	$\bowtie_{\mathcal{P}}$	2	Pairs each tuple from the left with every tuple of the right input relation if their shared attributes match and concatenates them.

The intersection $\mathcal{R}_L \cap \mathcal{R}_R$ generates a new relation of all tuples contained in $\mathcal{R}_L$ AND $\mathcal{R}_R$, as expressed by Equation (4.4).

$$\mathcal{R}_L \cup \mathcal{R}_R = \{t : t \in \mathcal{R}_L \wedge t \in \mathcal{R}_R\} \tag{4.4}$$

The difference $\mathcal{R}_L \setminus \mathcal{R}_R$ generates a new relation of all tuples contained in $\mathcal{R}_L$ AND NOT in $\mathcal{R}_R$, as expressed by Equation (4.5).

$$\mathcal{R}_L \setminus \mathcal{R}_R = \{t : t \in \mathcal{R}_L \wedge t \notin \mathcal{R}_R\} \tag{4.5}$$

These basic set operations simply combine two input relations without changing its structure, i.e., $\texttt{SCH}(R_I) = \texttt{SCH}(R_O)$. Due to relations being sets, duplicates resulting from a union operation are *implictly* eliminated.

4.2.2.2 Cartesian Product

The binary, *cartesian product* or *cross product* $\mathcal{R}_L \times \mathcal{R}_R$ of two input relations $\mathcal{R}_L, \mathcal{R}_R$ concatenates every tuple $t_L \in \mathcal{R}_L$ with every tuple $t_R \in \mathcal{R}_R$ to form a new output tuple, as expressed by Equation (4.6).

$$\mathcal{R}_L \times \mathcal{R}_R = \{(t_L, t_R) : t_L \in \mathcal{R}_L \wedge t_R \in \mathcal{R}_R\} \tag{4.6}$$

The result is a relation that contains all the attributes of $\mathcal{R}_L$ and $\mathcal{R}_R$, i.e., $\texttt{SCH}(\mathcal{R}_L \times \mathcal{R}_R) = \texttt{SCH}(\mathcal{R}_L) \cup \texttt{SCH}(\mathcal{R}_R)$ and every possible permutation of tuples from the input relations.

4.2.2.3 Rename

The unary *rename* operator $\rho_{\mathcal{A}_A \to \mathcal{A}_B}(\mathcal{R})$ renames the attribute $\mathcal{A}_A$ to $\mathcal{A}_B$ without changing any attribute values. This can be useful to elliminate collisions before applying the cartesian product or to enforce a natural join on differently named attributes.

4.2.2.4 Selection

The unary, generalized *selection* operator $\sigma_\phi(\mathcal{R})$ applied on an input relation $\mathcal{R}$ creates an output relation that contains a subset of tuples $t \in \mathcal{R}$ such that only tuples that match the predicate ϕ are retained as expressed by Equation (4.7).

$$\sigma_\phi(\mathcal{R}) = \{t \in \mathcal{R} : \phi(t)\} \subset \mathcal{R} \tag{4.7}$$

The predicate ϕ can be any conditional statement consisting of individual atoms that involve attributes of $\mathcal{R}$ or any constant value and comparison operators $=, \neq, >, <, \geq, \leq$. Individual atoms can also be combined by logical operators $\wedge$, $\vee$ or $\neg$. Examples could be $\mathcal{A}_1 \geq 2$ (to express that an attribute should be greater to a constant) or $\mathcal{A}_2 = \mathcal{A}_3$ (to express that an attribut must be equal to another attribute) or $\mathcal{A}_1 \geq 2 \wedge \mathcal{A}_2 = \mathcal{A}_3$ to combine the two with $A_1, A_2, A_3 \in \mathtt{SCH}(\mathcal{R})$.

4.2.2.5 Projection

The unary *projection* operator $\pi_\chi(\mathcal{R})$ with $\chi \subset \mathtt{SCH}(\mathcal{R})$ applied on an input relation $\mathcal{R}$ creates an output relation that only contains the attributes listed in χ, i.e., $\mathtt{SCH}(\pi_\chi(\mathcal{R})) = \chi$, as expressed by Equation (4.8).

$$\pi_\chi(\mathcal{R}) = \{t\,[\chi] : t \in \mathcal{R}\} \text{ with } t\,[\chi] = \{t[\mathcal{A}] : \mathcal{A} \in \chi\} \tag{4.8}$$

All the tuples in $\mathcal{R}$ are retained, however, resulting duplicates are removed.

4.2.2.6 Natural Join

The binary, *natural join* operator $\mathcal{R}_L \bowtie \mathcal{R}_R$ on two input relations $\mathcal{R}_L$, $\mathcal{R}_R$ concatenates every tuple $t_L \in \mathcal{R}_L$ with every tuple $t_R \in \mathcal{R}_R$ to form a new output tuple, if the attribute values of t_L and t_R are the same for the shared attribues $\xi = \{\mathcal{A} : \mathcal{A} \in \mathtt{SCH}(\mathcal{R}_L) \wedge \mathcal{A} \in \mathtt{SCH}(\mathcal{R}_R)\}$ as expressed by Equation (4.9)

$$\mathcal{R}_L \bowtie \mathcal{R}_R = \{(t_L, t_R) : t_L \in \mathcal{R}_L \wedge t_R \in \mathcal{R}_R \wedge t_L\,[\xi] = t_R\,[\xi]\} \tag{4.9}$$

The result is a relation that contains all the attributes of $\mathcal{R}_L$ and $\mathcal{R}_R$, i.e., $\texttt{SCH}(\mathcal{R}_L \times \mathcal{R}_R) = \texttt{SCH}(\mathcal{R}_L) \cup \texttt{SCH}(\mathcal{R}_R)$. If two relations coincide in all their attributes, the natural join becomes a cartesian product. Furthermore, the natural join can be expressed as a concatenation of the cartesian product with a selection. For example, given a relation $\mathcal{R}_L$ with $\texttt{SCH}(\mathcal{R}_L) = (\mathcal{A}_A, \mathcal{A}_B, \mathcal{A}_C)$ and R_R with $\texttt{SCH}(\mathcal{R}_R) = (\mathcal{A}_B, \mathcal{A}_D, \mathcal{A}_F)$, then $\mathcal{R}_L \bowtie \mathcal{R}_R \equiv \sigma_{\mathcal{A}_{L.B}=\mathcal{A}_{R.B}}(\mathcal{R}_L \times \mathcal{R}_R)$.

4.2.2.7 Expressing Queries

The following Example 4.3 illustrates how relational operators can be combined to form complex queries. Since expressing queries in such a way is quite inconvenient, in practice, queries are usually expressed through an intermediate query language which is then translated to the relational operators. A famous example of such a language in the domain of relational databases is SQL.

Example 4.3 Searching for Paintings Using Relational Algebra

The following tables lists the schema and extent of $\mathcal{R}_{\texttt{paintings}}$ and $\mathcal{R}_{\texttt{artists}}$.

$\mathcal{R}_{\texttt{painting}}$	$\mathcal{A}^*_{\texttt{title}}$	$\overline{\mathcal{A}}_{\texttt{artist}}$	$\mathcal{A}_{\texttt{painted}}$
t_1	Mona Lisa	Leonardo da Vinci	1506
t_2	The Starry Night	Vincent van Gogh	1889
t_3	Las Meninas	Diego Velázquez	1665

$\mathcal{R}_{\texttt{artist}}$	$\mathcal{A}^*_{\texttt{artist}}$	$\mathcal{A}_{\texttt{birth}}$	$\mathcal{A}_{\texttt{death}}$
t_1	Leonardo da Vinci	1452	1519
t_2	Vincent van Gogh	1853	1890
t_3	Diego Velázquez	1599	1660

Using relational algebra, the query "return the names of all paintings that were painted by an artist who died after 1800" can be expressed by joining $\mathcal{R}_{\texttt{paintings}}$ and $\mathcal{R}_{\texttt{artists}}$, followed by a selection and projection:

$$\mathcal{R}_{\texttt{result}} = \pi_{A_{\texttt{title}}}(\sigma_{A_{\texttt{death}} \geq 1800}(\mathcal{R}_{\texttt{paintings}} \bowtie \mathcal{R}_{\texttt{artists}}))$$

Execution of this query leads to the following relation $\mathcal{R}_{\texttt{result}}$.

$\mathcal{R}_{\texttt{result}}$	$\mathcal{A}^*_{\texttt{title}}$
t_2	The Starry Night

4.2.3 Extensions

While the relational model and the relational algebra forms the foundation of many modern DBMS, the model as originally proposed by Codd has often turned out to be too limited to accomodate certain functionality as required, for example, by the SQL standard [Cha12; GUW09]. For example, many applications require storage of duplicate data and therefore, the notion of a relation – which is a set of tuples and thus does not allow for duplicates – is inadequate. Over the years, this has led to a growing list of proposals for extensions, some of which have seen adoption while other's have remained theoretical in nature.

4.2.3.1 Relations vs. Bag vs. Sequences

4.2.3.2 Extended Projection

4.2.3.3 Ranked Relational Algebra

4.2.3.4 Similarity Search and the Relational Model

Using the relationship between (diss-)similarity and distance, it has been shown by Giangreco et al., that for a database system to be able to support similarity search given the relational model for databases as described in Section 4.2, one can extend the set system of allowed data domains $\mathbb{D}$ by $\mathcal{D} \subset \mathbb{R}^{dim}, dim \in \mathbb{N}_{>1}$ and postulate the existence of a relational similarity operator $\tau_{\delta(\cdot,\cdot),a,q}(R)$ that *"performs a similarity query under a distance $\delta(\cdot,\cdot)$ applied on an attribute a of relation R and compared to a query vector q."* ([Gia18], p. 138). Such an operation introduces an implicit attribute in the underlying relation $\mathcal{R}$, which in turn induces an ascending ordering of the tuples. Using this operation, one can go on to define two concrete implementations, namely $\tau^{kNN}_{\delta(\cdot,\cdot),a,q}(\mathcal{R})$, and $\tau^{\epsilon NN}_{\delta(\cdot,\cdot),a,q}(\mathcal{R})$, which limit the number of retrieved results by their cardinality k or a maximum cut-off distance ϵ respectively.

While postulating a new, relational operation $\tau^{kNN}_{\delta(\cdot,\cdot),a,q}(\mathcal{R})$ or $\tau^{\epsilon NN}_{\delta(\cdot,\cdot),a,q}(\mathcal{R})$ as proposed by [Gia18] has a certain elegance to it, it comes with limitations that become apparent upon disection of its structure. In its postulated form, τ addresses several functions at once:

1. It specifies the distance function that should be evaluated.
2. It generates an implicit distance attribute on the underlying relation $\mathcal{R}$.
3. It imposes an ascending ordering and limits the cardinality of $\mathcal{R}$ based on a predicate or a specified limit.

While being very specific and thus straightforward to implement, the amalgamation of all this functionality into a single operation is very specifically tailored to the use-case of similarity search and only of limited value when considering more general proximity-based query operations. If one would, for example, want to obtain the k farthest neighbours rather than the k nearest neighbours, as necessary when doing MIPS or obtaining negative examples, we would have to either change the distance function or extend the definition of τ.

Another important issue with the definition of τ in its current form is that despite the two operations $\tau^{kNN}_{\delta(\cdot,\cdot),a,q}(\mathcal{R})$ and $\tau^{\epsilon NN}_{\delta(\cdot,\cdot),a,q}(\mathcal{R})$ serving a very similar purpose, they behave very differently with respect to other operations. Generally, $\tau^{kNN}_{\delta(\cdot,\cdot),a,q}(\mathcal{R})$ does not commute with any selection σ due to the inherent limiting of the cardinality to a constant value, hence:

$$\sigma(\tau^{kNN}_{\delta(\cdot,\cdot),a,q}(\mathcal{R})) \neq \tau^{kNN}_{\delta(\cdot,\cdot),a,q}(\sigma(\mathcal{R})) \tag{4.10}$$

The left-hand side of Equation (4.10) filters the results of a kNN-search on $\mathcal{R}$, thus returning $n \leq k$ results, wherein $n = k$ only if σ matches all tuples. The right-hand side of Equation (4.10) performs a kNN-search on a pre-filtered relation $\mathcal{R}$, also returning $n \leq k$ entries. However, n will only be smaller than k if σ selects fewer than k tuples.

The same isn't true for $\tau^{\epsilon NN}_{\delta(\cdot,\cdot),a,q}(\mathcal{R})$, due to the limitation of the cardinality being facilitated by an *implicit* selection $\sigma_{\delta \leq \epsilon}$.

$$\sigma(\tau^{\epsilon NN}_{\delta(\cdot,\cdot),a,q}(\mathcal{R})) = \tau^{\epsilon NN}_{\delta(\cdot,\cdot),a,q}(\sigma(\mathcal{R})) \tag{4.11}$$

Hence, σ and $\tau^{\epsilon NN}_{\delta(\cdot,\cdot),a,q}(\mathcal{R})$ commute and yield equivalent results as long as σ does not rely on the distance attribute introduced by $\tau^{\epsilon NN}_{\delta(\cdot,\cdot),a,q}(\sigma(\mathcal{R}))$

4.3 Queries: From Expression to Execution

4.3.1 Structured Query Language (SQL)

4.3.2 Query Planning

4.3.3 Query Execution

4.4 Storage, Indexes and Caching

4.5 Architectual Considerations

Part III

Dynamic Multimedia Data Management

5

Modelling a Database for Dynamic Multimedia Data

5.1 Generalized Distance Based Operations

Starting with the metric space model for similarity search [ZAD+06] presented in Chapter 3, we propose to extend the notion of proximity-based similarity search to that of a more general *proximity based query* following Definition 5.1.

Definition 5.1 Proximity Based Query

Any database query operation on relation $\mathcal{R}$ that relies on the evaluation of a distance function $\delta(a_i, q)$ that calculates the distance between an attribute $a_i \in \mathcal{D}_q$ and some query $q \in \mathcal{D}_q$ given $\mathcal{D}_q \in \texttt{SCH}(\mathcal{R})$ and distance function $\delta : \mathcal{D}_q \times \mathcal{D}_q \rightarrow \mathbb{R}$, is called a *proximity based query*. We call q the *query* and a_i the *probing attribute* both sharing the same *query data domain* $\mathcal{D}_q$.

It is important to note, that Definition 5.1 simply requires a notion of proximity between some attribute and a query to be obtained by evaluating some distance function. The definition does not make any assumption as to what data domains $\mathcal{D}_q$ query and probing attribute belong to nor how the distance is being used within the query, once it has been obtained.

We observe, that similarity search falls into the broader category of a proximity based queries, wherein the distance is used to rank the relation and subsequently limit its cardinality. In addition, proximity based queries can host operations such as evaluating a distance function followed by some filtering or grouping based on the computed value, i.e, we are not limited to mere search.

5.1.1 Revisiting Distance Computation

Since the choice of the distance function δ is a crucial part of any proximity based query, it is worth revisiting its definition. Again, starting with the metric space modell [ZAD+06], we identify the following (implicit) constraints with respect to the distance function:

1. The domain (i.e., the input) of the distance function δ is assumed to be $\mathbb{R}^{dim} \times \mathbb{R}^{dim}$, that is, the distance function is assumed to be a binary function and arguments are restricted to real-valued vectors.

2. The codomain (i.e., the output) of the distance function δ is assumed to be $\mathbb{R}_{\geq 0}$, hence, the generated distance value is a positive, real number.

3. The pair $(\mathcal{D}_q, \delta)$ constitutes a metric, thus satisfying the non-negativity, identity of indiscernibles, symmetry and subadditivity properties.

Upon closer examination, one can come to the conclusion that there is good reason to assume the codomain of δ to lie in $\mathbb{R}$. On the one hand, it is obviously convenient both for the underlying mathematics as well as from an implementation prespective. More importantly, however, real numbers – unlike, for example, complex numbers or vectors – come with a natural, total ordering, which is required for the sorting that is part of many proximity based queries. If, however, we turn to Example 5.1, we realise that it is not reasonable to impose a general restriction of the domain of the distance function to $\mathbb{R}^{dim}$ with $dim \in \mathbb{N}^+$

Example 5.1 Maximum Inner Product Search for MRF

In MRF (see Section 2.3), we try to obtain the signal vector $a_{i \in \mathbb{N}} \in \mathcal{D}_q \subset \mathbb{C}^{dim}$ so that it maximizes the inner product to a signal (query) vector $q \in \mathbb{C}^{dim}$. In this case, the distance function δ has the form $\delta \colon \mathbb{C}^{dim} \times \mathbb{C}^{dim} \to \mathbb{R}_{\geq 0}$. Hence, the domain of δ is a complex vector space.

Obviously, this limitation of the definition of δ can be easily remediated simply by extending the set of supported data domains $\mathbb{D}$ by $\mathbb{C}^{dim}$ similarily to how we did for $\mathbb{R}^{dim}$ as proposed by [Gia18]. Nevertheless, we have to acknowledge, that the text-book definition of a distance function is obviously too limited for some real-world applications, and that the query and probing arguments of a distance function could be any type of value supported by the DBMS. Another

example could be the *Levenshtein distance* [Lev65] – often used in computer linguistics – which is a distance metric on string values.

If now in addition, we consider Example 5.2, we see that restricting oneself to binary functions is also too limiting when facing certain practical scenarios.

Example 5.2 Distance Between a Vector and a Hyperplane

To find positive and negative examples $a_i \in \mathcal{D}_q \subset \mathbb{R}^{dim}$ given a linear classifier, e.g., provided by a SVM, we evaluate the distances between the attributes and a (query-)hyperplane described as $\mathbf{q}^T\mathbf{x} - b = 0$ with $\mathbf{q}, \mathbf{x} \in \mathbb{R}^{dim}$ and $b \in \mathbb{R}$. The distance function is then given by:

$$\delta(\mathbf{a}_i, \mathbf{q}, b) = \frac{\left\|\mathbf{q}^T\mathbf{a_i} + b\right\|}{\left\|\mathbf{q}\right\|} \tag{5.1}$$

δ now has the form $\delta\colon \mathbb{R}^{dim} \times \mathbb{R}^{dim} \times \mathbb{R} \to \mathbb{R}$. Hence, δ is no longer a binary but a ternay function with arguments $\mathbf{a_i}$, $\mathbf{q}$ and b. Furthermore, the distance may take negative values depending on whether a result is considered a positive or negative example.

Again, we are confronted with an example that violates the text-book definition of a distance function. While the idealized idea that distance functions used in proximity based queries must always constitute a metric on some real-valued vector space may be very common [ZAD+06], we see in practice that this assumption is often violated [BS19] and that many functions used to calculate proximity between objects, are not actual metrics. Even though, this can be a disadvantage when considering high-dimensional index structures that exploit the geometric properties of metric functions, it is obvious that such functions exist and must thus be considered in any generalized database application supporting proximity based queries.

More examples

To summarize, we now find ourselves in the position where, on the one hand, we require a proper definition of what is considered to be a valid distance function so as to be able to efficiently plan and execute queries using them, while on the other hand keeping that definition open enough to accomodate the many different, practical applications now and in the future.

In order to address the aforementioned limitations while still accomodating classical, metric distance functions and the need to be able to plan query execution, we propose the extension of a the distance function to the more general notion of a Distance Function Class (DFC) following Definition 5.2.

Definition 5.2 Distance Function Class

A DFC $\hat{\delta}\colon \mathcal{D}_q \times \mathcal{D}_q \times \mathcal{D}_1 \ldots \times \mathcal{D}_n \to \mathbb{R}$ is an n-ary but at least binary function $\hat{\delta}(p, q, s_1, \ldots, s_n)$ that outputs a distance $d \in \mathbb{R}$ between a probing argument $p \in \mathcal{D}_p$ and a query argument $q \in \mathcal{D}_q$ using a defined number of *support arguments* $s_{k\in\mathbb{N}}$ from any of the data domains in $\mathbb{D}$.

A N-ary DFC is uniquely defined by its *signature*, a $(N+1)$-tuple specifying the name of the DFC and the data domains $\mathcal{D}_i$ of the arguments it can accept as defined in Equation (5.2). As a convention for notation, the probing argument p of a DFC $\hat{\delta}(p, q, s_1, ..., s_n)$ will always come first, followed by the query argument q, again followed by its support arguments s_k in no particular order.

$$\texttt{SIG}(\hat{\delta}) = (\texttt{NAME}, \mathcal{D}_p, \mathcal{D}_q, \mathcal{D}_1, ...\mathcal{D}_{N-2}) \tag{5.2}$$

Furthermore, in the context a physical execution plan for a proximity based query, we require the query argument q and the support arguments s_i for a DFC to remain constant. Hence, irrespective of the arity of the DFC, its *implementation* δ is always a unary function with signature $(\texttt{NAME}, \mathcal{D}_q)$, i.e., Equation (5.3) holds.

$$\delta = \texttt{IMP}(\hat{\delta}) \ni \texttt{SIG}(\delta) = \texttt{SIG}(\texttt{IMP}(\hat{\delta})) = (\texttt{NAME}, \mathcal{D}_q) \,\forall\, \hat{\delta} \tag{5.3}$$

With the idea of a DFC and its implementation in mind, we can start to reason about their properties in the broader context of database operations in general and proximity based queries in particular:

DFC and Proximity Based Queries In extension to Definition 5.1, we realise that any logical or physical execution plan for a database query that involves the evaluation of a DFC or its implementation falls into the category of a proximity based query.

Purity of Domain Since the purpose of a DFC is to quantify proximity between a probing argument a and a query argument q, we can assume that $a, q \in \mathcal{D}_q$, i.e., both arguments usually belong to the same data domain $\mathcal{D}_q \in \mathbb{D}$.

Scalar Codomain A DFC and its implementations are a scalar functions, that is, for any choice of arguments they produce a single, scalar output $d \in \mathbb{R}$.

From the perspective of the DBMS implementation, DFCs represent a family of higher-order functions, which in turn parametrize unary functions using the query q and support arguments s_k. Implementing a DFC in essence means

chosing values for q and s_k in the context of a query. This implementation requirement enshrines the fact that for a proximity based query, altering q and s_k effectively gives rise to a new query, which is why for a given execution plan instance, those values must remain constant.

During the execution of a proximity based query, the DBMS can simply read the probing arguments from the data store for every tuple in the underlying relation $\mathcal{R}$, execute the DFC's implementation using the probing argument and obtain the output of the DFC.

A very simple yet widely used example of a DFC would be the *Minkowski Distance* $\hat{\delta}_M \colon \mathbb{R}^{dim} \times \mathbb{R}^{dim} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$ with its two, partial implementations, the Manhattan (L1) and the Euclidean (L2) distance:

$$\hat{\delta}_M(\mathbf{a}, \mathbf{q}, p) = \left(\sum_{i=1}^{n} \mid q_i - a_i \mid^p \right)^{\frac{1}{p}} \tag{5.4}$$

$$\delta_{L1}(\hat{\mathbf{a}}, \mathbf{q}) = \mathtt{IMP}_{p=1}(\hat{\delta}_M) = \sum_{i=1}^{n} \mid q_i - a_i \mid \tag{5.5}$$

$$\delta_{L2}(\hat{\mathbf{a}}, \mathbf{q}) = \mathtt{IMP}_{p=2}(\hat{\delta}_M) = \sqrt{\sum_{i=1}^{n} \mid q_i - a_i \mid^2} \tag{5.6}$$

However, using the idea of a DFC, we can express many different types of much more complex distance functions in a simple yet powerful framework. Two examples are illustrated in Figure 5.1 and it is easy to see that both the functions used in Example 5.1 (complex domain) and Example 5.2 (ternay function with scalar support argument) fall into the category of a DFC. We will show in Section 5.1.3, that a DBMS can leverage the *signature* and the *implementation* property of a DFC to plan the execution of proximity based queries.

As an aside, we must address the role of dimensionality *dim* in the case of data domains that are subsets of vector spaces, i.e., $\mathcal{D}_q \subset \mathbb{R}^{dim}$ or $\mathcal{D}_q \subset \mathbb{C}^{dim}$. One could argue, that the dimensionality of such a vector space can also be seen as a parameter of the DFC. Irrespective of the merits such an argument might have, we consider the dimensionality of the vector space to be a strictly structural property of the underlying data domain $\mathcal{D}_q$ and thus thightly coupled to the data type. This means, that dimensionality as well as the data type are well-defined and remain constant within and, for most parts, even between queries for a given relation.

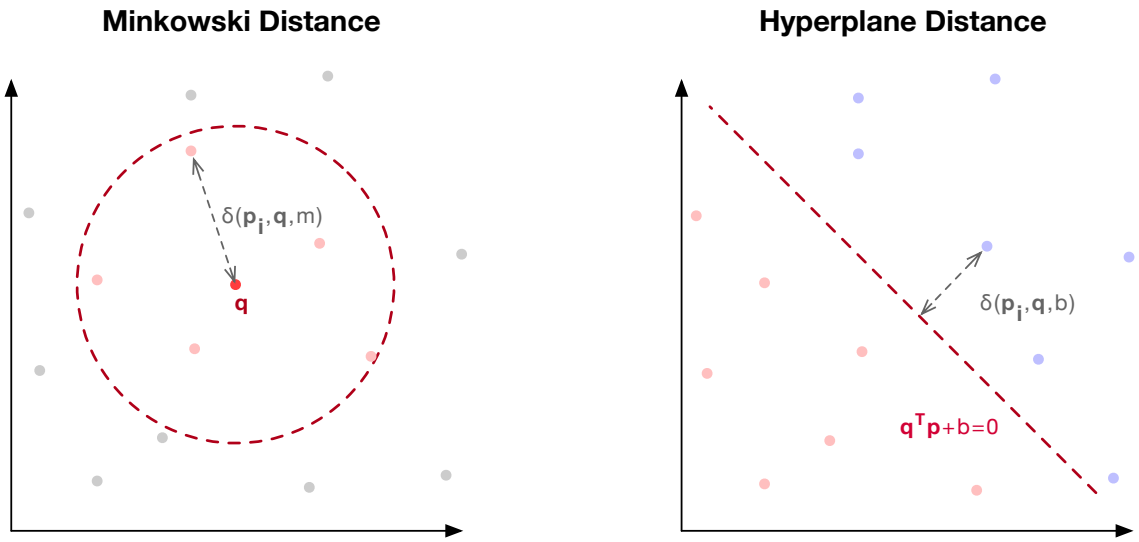

Figure 5.1 Two examples of distance functions that fall into the broader category of a DFC. On the left, the point-distance between q and p_i and on the right, the distance between a hyperplane $q^T x + b = 0$ and points p_i.

5.1.2 Extending the Relational Model

Using Definition 5.2, one can start to integrate DFC into the relational data model. Following [Gia18], we first assume $\mathbb{D}$ – the set system of data domains supported by the database – to be extended by whatever data domain $\mathcal{D}_q$ is required. Following that recipe, a DBMS can be extended to support every type of (mathematical) object that is useful for a given application, such as but not limited to $\mathbb{R}^{dim}$ or $\mathbb{C}^{dim}$.

We now use the idea of an extended projection $\pi_{\mathcal{E}}$ – as proposed by [GHQ95; GUW09] and introduced in Section 4.2 – wherein $\mathcal{E}$ is simply a list of expressions involving attributes $\mathcal{A}_i \in \mathcal{R}$. That is, in addition to the simple projection onto attributes $\mathcal{A}_i$, an extended projection may also include the evaluation of algebraic and function expressions. Assuming that a function invocation involving constants as well as arguments from the underlying relation are supported, the evaluation of a DFC can be expressed as follows:

Definition 5.3 Distance Function Class in Extended Projection $\pi_{\mathcal{E}}$

Let $\hat{\delta}\colon \mathcal{D}_q \times \mathcal{D}_q \times \mathcal{D}_1 ... \times \mathcal{D}_{N-2} \to \mathbb{R}$ be a N-ary DFC and $\delta = \texttt{IMP}(\hat{\delta})$ its implementation. Let further $\mathcal{R}$ be a relation with attribute $\mathcal{A}_p \in \texttt{SCH}(\mathcal{R})$. The *extended projection* $\pi_{\delta(\mathcal{A}_p)}(\mathcal{R})$ describes the evaluation of δ using argument values $p_i \in \mathcal{A}_p$ as probing arguments. Applying $\pi_{\delta(\mathcal{A}_p)}$ on a M-ary relation $\mathcal{R}$ introduces a new distance attribute $\mathcal{A}_d$, i.e., $\texttt{SCH}(\pi_{\delta(\mathcal{A}_p),\mathcal{A}_1,...,\mathcal{A}_M}(\mathcal{R})) = \texttt{SCH}(\mathcal{R}) \cup \{\mathcal{A}_d\}$

Obviously, the evaluation of multiple DFCs in a single, extended projection or the combination of simple attribute projections with the evaluation of DFCs are also allowed. In fact, all of the following expressions are valid examples of the extended projection on relation $\mathcal{R}$ with $\texttt{SCH}(\mathcal{R}) = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4\}$ and $\delta_i = \texttt{IMP}(\hat{\delta}_i)$: (i) Evaluation of multiple DFCs with same or different probing arguments (e.g., $\pi_{\delta_1(\mathcal{A}_1),\delta_2(\mathcal{A}_1)}(\mathcal{R})$), (ii) evaluation of same DFC using same or different probing arguments (e.g., $\pi_{\delta_1(\mathcal{A}_1),\delta_1(\mathcal{A}_2)}(\mathcal{R})$), (iii) combination of DFCs with any attribute projection and/or algebraic expression evaluation (e.g., $\pi_{\delta(\mathcal{A}_1),\mathcal{A}_2,\mathcal{A}_4}(\mathcal{R})$).

In order to support sorting by (the obtained) distance, which is an essential part of many types of proximity based queries, we use the idea of a *ranked relation* $\mathcal{R}^O$, as proposed by [LCI+05] and described in Section 4.2.3. A ranked relation [1] $\mathcal{R}^O$ exhibits an ordering of tuples $t_i \in \mathcal{R}^O$, which is induced by the ranking or scoring predicate O. As opposed to [LCI+05] and more in line with [GUW09], we assume O to be a simple sequence of pairs $(\mathcal{A}_i, \texttt{DIR})$ with $\texttt{DIR} \in \{\uparrow, \downarrow\}$ and $\mathcal{A}_i \in \mathcal{R}^O$ each denoting an attribute to order by and the desired sort direction *ascending* or *descending*[2]. If $\mathcal{R}$ does not exhibit a specific ranking, then the O is simply ommitted.

In order to sort an (unordered) relation, we introduce the relational operator τ_O such that:

$$\tau_O(\mathcal{R}) = \mathcal{R}^O \tag{5.7}$$

Again using the relation $\mathcal{R}$ with $\texttt{SCH}(\mathcal{R}) = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4\}$, the application of $\tau_{((\mathcal{A}_1,\uparrow),(\mathcal{A}_2,\downarrow))}$ on relation $\mathcal{R}$ sorts the tuples $t_i \in \mathcal{R}$ based on $\mathcal{A}_1$ in ascending order and breaks ties sorting by $\mathcal{A}_2$ in descending order. Tuples that are equal in both attributes appear in an arbitrary order.

So as to be able to execute a specific type of proximity based query – namely, kFN or kNN – we require one last extension to the relational algebra in the form of the k-selection operator $\lambda_k(\mathcal{R})$. The $\lambda_k(\mathcal{R})$ operator simply limits the cardinality of a relation $\mathcal{R}$ to k tuples, without changing the order of a ranked relation $\mathcal{R}^O$. It is trivial to see that for a ranked relation $\mathcal{R}^O$, $\lambda_k(\mathcal{R}^O)$ limits the sequence of tuples to the top k results w.r.t. to the ordering induced by O.

We can now use the example presented in Example 5.3 to demonstrate the flexibility of the proposed algebra extensions and the different types of queries that can be expressed through $\pi_\mathcal{E}$, τ_O, λ_k.

[1] Mathematically speaking, ranked relations are not longer relations rather than sequences.

[2] We argue that formally, the evaluation of a *scoring function*, as proposed by [LCI+05], can be expressed within the scope of the extended projection and does not need to be part of the order operation.

Example 5.3 Relation listing paintings with feature vectors

$\mathcal{R}_p$ lists paintings with their title $\mathcal{A}_t$, the year of their creation $\mathcal{A}_y$ and some arbitrary feature vector $\mathcal{A}_f$. Note that $\mathcal{D}_f \subset \mathbb{R}^3$.

$\mathcal{R}_p$	$\mathcal{A}_t$	$\mathcal{A}_y$	$\mathcal{A}_f$
t_1	Mona Lisa	1506	$[0.0, 0.2, -1.3]$
t_2	The Starry Night	1889	$[1.0, 0.9, 2.6]$
...	...	...	...
t_N	Las Meninas	1665	$[-0.5, 3.0, 0.8]$

Using DFCs and the proposed, relational algebra, we can now express:

Name	Result	Algebraic Form
NNS / kNN	$\mathcal{R}^{(\mathcal{A}_d,\uparrow)}$	$\lambda_k(\tau_{(\mathcal{A}_d,\uparrow)}(\pi_{\mathcal{A}_y,\delta(\mathcal{A}_f)}(\mathcal{R}_p)))$
FNS / kFN	$\mathcal{R}^{(\mathcal{A}_d,\downarrow)}$	$\lambda_k(\tau_{(\mathcal{A}_d,\downarrow)}(\pi_{\mathcal{A}_y,\delta(\mathcal{A}_f)}(\mathcal{R}_p)))$
ϵNN	$\mathcal{R}^{(\mathcal{A}_d,\uparrow)}$	$\tau_{(\mathcal{A}_d,\uparrow)}(\sigma_{\mathcal{A}_d \leq \epsilon}(\pi_{\mathcal{A}_y,\delta(\mathcal{A}_f)}(\mathcal{R}_p)))$
NNS w. selection	$\mathcal{R}^{(\mathcal{A}_d,\uparrow)}$	$\tau_{(\mathcal{A}_d,\uparrow)}(\pi_{\mathcal{A}_{year},\delta(\mathcal{A}_f)}(\sigma_{\mathcal{A}_y=1889}(\mathcal{R}_p)))$
Multi order	$\mathcal{R}^{(\mathcal{A}_{d1},\uparrow),(\mathcal{A}_{d2},\uparrow)}$	$\tau_{(\mathcal{A}_{d1},\uparrow),(\mathcal{A}_{d2},\uparrow)}(\pi_{\delta_1(\mathcal{A}_f),\delta_2(\mathcal{A}_f)}(\mathcal{R}_p))$
DFS & aggregate	$\mathcal{R}$	

5.1.2.1 Algebraic Properties of π, τ_O, λ_k

During the logical optimization phase of query planning, a DBMS leverages the algebraic properties of the operators it employs in an attempt to generate a more efficient execution plan. Most importantly, these properties determine the types of operations that can be permuted or even exchanged. As opposed to classical relational algebra, we must consider: (i) Whether a permutation

Elaborate on algebraic properties of π, τ_O, λ_k

5.1.2.2 Combination with Other Extensions

Elaborate on how π, τ_O, λ_k can be combined with other relational extensions

5.1.3 DFCs and Query Planning

The requirements imposed on the structure of a DFC can be leveraged to enable a DBMS to plan and optimize the execution of a proximity based query by not just altering the execution plan in terms of (relational) operators but also in

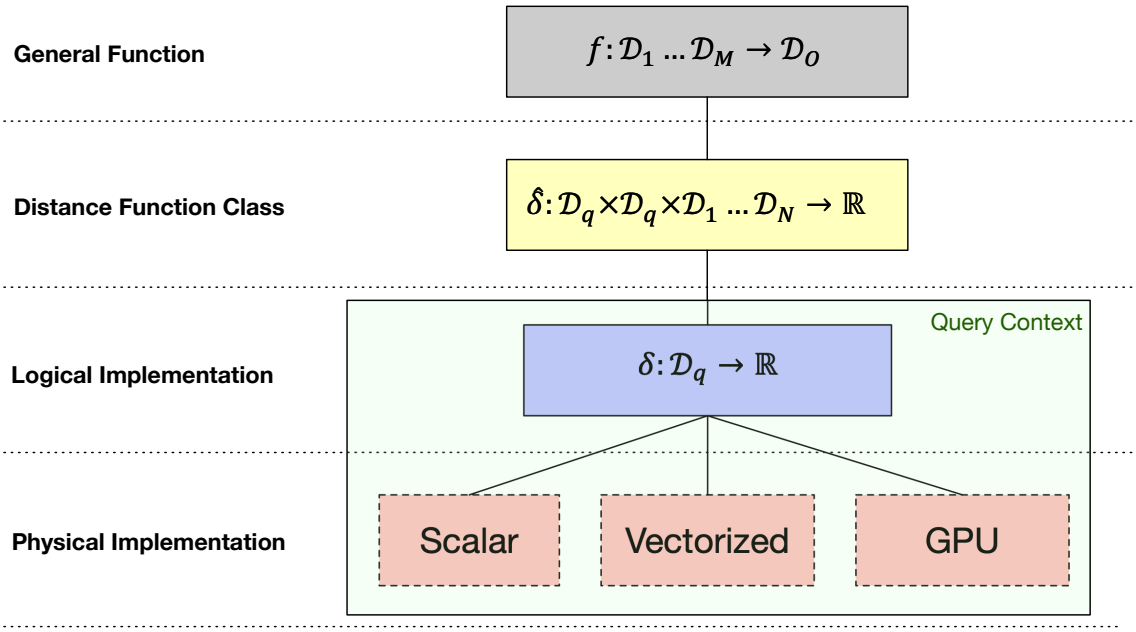

Figure 5.2 Two examples of distance functions that fall into the broader category of a DFC. On the left, the point-distance between q and p_i and on the right, the distance between a hyperplane $q^T x + b = 0$ and points p_i.

terms of DFC implementations that should be employed. To illustrate this, we proposed the notion of the *function hierarchy* depicted in 5.2.

The figure depicts executable functions from a perspective of the DBMS and the amount of insights it has about them. At the very top, we see general functions such as UDFs. In general, the DBMS has little to no knowledge about the internals of these functions and can thus only provide very limited support for optimizing their execution. As we move down the hierarchy, the DBMS "gains" insights with respect to the function's properties. At the level of the DFC, the DBMS is aware of the function's name and the general structure of the DFCs domain and codomain.

Elaborate on how the structure of a DFC can be exploited for query planning

5.2 Cost Model for Retrieval Accuracy

Describe cost model for execution plans with following properties:

- Cost as a function of atomic costs: $f(a_{cpu}, a_{io}, a_{memory}, a_{accuracy}) \longrightarrow C$
- Means to estimate results accuracy and associated considerations from execution path (e.g., when using index) based on properties of the index

– Means to specify importance of accurate results (e.g., global, per-query, context-based i.e. when doing 1NN search) in comparison to other factors

– Systems perspective 1: How can such a cost model be applied during query planning and optimization?

5.3 Adaptive Index Management

Figure 5.3 Adaptive index structures overview.

Describe model for index management in the face of changing data (adaptive index management):

– Reason about properties of secondary indexes for NNS (e.q., PQ, VA, LSH) with regards to data change

– Derivation of error bounds possible (e.g., usable for planning)?! Use in query planning?

– Systems perspective 1: How to cope with "dirty" indexes? Proposal: hot vs. cold index, auxilary data structure, offline optimization, see Figure 5.3

– Systems perspective 2: On-demand index based on query workload?

5.4 Architecture Model

Putting everything together into a unified systems model (base on previous work + aforementioned aspects).

6

Cottontail DB

Implementation chapter for Cottontail DB

Part IV

Discussion

7

Evaluation

7.1 Interactive Multimedia Retrieval

vitrivr's participation to VBS, LSC etc, DRES.

7.2 Adaptive Index Management

Brute force vs. plain index vs. index with auxilary data structure

7.3 Cost Model

Benchmark effect of cost model in different settings (e.g. based on use cases from chapter 2)

8

Related Work

Publications related to this thesis, i.e., because they may be similar. No foundations!

9

Conclusion & Future Work

Appendix

Bibliography

[ABC+76] M. M. Astrahan, M. W. Blasgen, D. D. Chamberlin, K. P. Eswaran, J. N. Gray, P. P. Griffiths, W. F. King, R. A. Lorie, P. R. McJones, J. W. Mehl, G. R. Putzolu, I. L. Traiger, B. W. Wade, and V. Watson. System r: relational approach to database management. *ACM Transactions on Database Systems*, 1(2):97–137, June 1976. ISSN: 0362-5915. DOI: 10.1145/320455.320457.

[BTVG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: speeded Up Robust Features. In *European Conference on Computer Vision*, pages 404–417, 2006.

[BS19] David Bernhauer and Tomáš Skopal. Non-metric similarity search using genetic TriGen. In Giuseppe Amato, Claudio Gennaro, Vincent Oria, and Miloš Radovanović, editors, *Similarity Search and Applications*, pages 86–93, Cham. Springer International Publishing, 2019. ISBN: 978-3-030-32047-8.

[BdB+07] Henk M Blanken, Arjen P de Vries, Henk Ernst Blok, and Ling Feng. *Multimedia Retrieval*. Springer, 2007.

[BGS+20] Samuel Börlin, Ralph Gasser, Florian Spiess, and Heiko Schuldt. 3D Model Retrieval Using Constructive Solid Geometry in Virtual Reality. In *Proceedings of the 2020 IEEE International Conference on Artificial Intelligence and Virtual Reality*, AIVR 2021, pages 373–374, 2020.

[CWW+10] Yang Cao, Hai Wang, Changhu Wang, Zhiwei Li, Liqing Zhang, and Lei Zhang. MindFinder: interactive sketch-based image search on millions of images. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, pages 1605–1608, New York, NY, USA. Association for Computing Machinery, 2010. ISBN: 978-1-60558-933-6. DOI: 10.1145/1873951.1874299.

[Cha12] Donald D. Chamberlin. Early History of SQL. *IEEE Annals of the History of Computing*, 34(4):78–82, 2012. DOI: 10.1109/MAHC.2012.61.

[Che76] Peter Pin-Shan Chen. The Entity-Relationship Model — Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9–36, March 1976. ISSN: 0362-5915. DOI: 10.1145/320434.320440.

[CTW+10] Nancy A. Chinchor, James J. Thomas, Pak Chung Wong, Michael G. Christel, and William Ribarsky. Multimedia Analysis + Visual Analytics = Multimedia Analytics. *IEEE Computer Graphics and Applications*, 30(5):52–60, 2010. DOI: 10.1109/MCG.2010.92.

[Cod70] E. F. Codd. Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 13(6):377–387, June 1970.

[GUW09] Hector Garcia-Molina, Jeffrey D. Ullmann, and Jennifer Widom. *DATABASE SYSTEMS The Complete Book, 2nd Edition*. second edition, 2009. ISBN: l.

[GRH+20] Ralph Gasser, Luca Rossetto, Silvan Heller, and Heiko Schuldt. Cottontail DB: an Open Source Database System for Multimedia Retrieval and Analysis. In *Proceedings of the 28th ACM International Conference on Multimedia*, ACM MM 2020, pages 4465–4468, 2020.

[GRS19a] Ralph Gasser, Luca Rossetto, and Heiko Schuldt. Multimodal Multimedia Retrieval with vitrivr. In *Proceedings of the 2019 International Conference on Multimedia Retrieval (ICMR 2019)*, ICMR 2019, pages 391–394, New York, NY, USA. Association for Computing Machinery, 2019. ISBN: 978-1-4503-6765-3. DOI: 10.1145/3323873.3326921.

[GRS19b] Ralph Gasser, Luca Rossetto, and Heiko Schuldt. Towards an all-purpose content-based multimedia information retrieval system. *CoRR*, abs/1902.03878, 2019.

[GLC+95] Asif Ghias, Jonathan Logan, David Chamberlin, and Brian C Smith. Query by Humming: musical Information Retrieval in an Audio Database. In *Proceedings of the Third ACM International Conference on Multimedia*, pages 231–236, 1995.

[Gia18] Ivan Giangreco. *Database Support for Large-Scale Multimedia Retrieval*. PhD thesis, University of Basel, Switzerland, August 2018.

[GS16] Ivan Giangreco and Heiko Schuldt. ADAMpro: database support for big multimedia retrieval. *Datenbank-Spektrum*, 16(1):17–26, 2016. DOI: 10.1007/s13222-015-0209-y.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[GHQ95] Ashish Kumar Gupta, Venky Harinarayan, and Dallan Quass. Generalized Projections: a Powerful Approach to Aggregation. In 1995.

[HR83] Theo Haerder and Andreas Reuter. Principles of transaction-oriented database recovery. *ACM Computing Surveys (CSUR)*, 15(4):287–317, 1983.

[HGI+21] Silvan Heller, Ralph Gasser, Cristina Illi, Maurizio Pasquinelli, Loris Sauter, Florian Spiess, and Heiko Schuldt. Towards Explainable Interactive Multi-Modal Video Retrieval with vitrivr. In *Proceedings of the 27th International Conference on Multimedia Modeling*, MMM 2021, pages 435–440, Cham. Springer International Publishing, 2021. ISBN: 978-3-030-67835-7. DOI: 10.1007/978-3-030-67835-7_41.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate Nearest Neighbors: towards Removing the Curse of Dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 604–613, 1998.

[JDS11] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product Quantization for Nearest Neighbor Search. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(1):117–128, January 2011. DOI: 10.1109/TPAMI.2010.57.

[JWZ+16] Björn Þór Jónsson, Marcel Worring, Jan Zahálka, Stevan Rudinac, and Laurent Amsaleg. Ten Research Questions for Scalable Multimedia Analytics. In *International Conference on Multimedia Modeling*, pages 290–302, 2016.

[KKE+10] Daniel Keim, Jörn Kohlhammer, Geoffrey Ellis, and Florian Mansmann. Mastering the information age: solving problems with visual analytics, 2010.

[Lev65] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics. Doklady*, 10:707–710, 1965.

[LCI+05] Chengkai Li, Kevin Chen-Chuan Chang, Ihab F. Ilyas, and Sumin Song. RankSQL: query Algebra and Optimization for Relational Top-k Queries. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, pages 131–142, New York, NY, USA. Association for Computing Machinery, 2005. ISBN: 1-59593-060-4. DOI: 10.1145/1066157.1066173.

[LKM+19] Jakub Lokoč, Gregor Kovalčík, Bernd Münzer, Klaus Schöffmann, Werner Bailer, Ralph Gasser, Stefanos Vrochidis, Phuong Anh Nguyen, Sitapa Rujikietgumjorn, and Kai Uwe Barthel. Interactive Search or Sequential Browsing? a Detailed Analysis of the Video Browser Showdown 2018. *ACM Transactions on Multimedia Computing, Communications, and Applications*. TOMM, 15(1):1–18, 2019.

[Low99] David G Lowe. Object Recognition from Local Scale-Invariant Features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157, 1999.

[Pet19] Alex Petrov. *Database Internals*. O'Reilly Media, Inc., 2019. ISBN: 978-1-4920-4034-7.

[Ros18] Luca Rossetto. *Multi-Modal Video Retrieval*. PhD thesis, University of Basel, Switzerland, September 2018.

[RAPG+19] Luca Rossetto, Mahnaz Amiri Parian, Ralph Gasser, Ivan Giangreco, Silvan Heller, and Heiko Schuldt. Deep learning-based concept detection in vitrivr. In *Proceedings of the 25th International Conference on Multimedia Modelling*, MMM 2019, pages 616–621, Cham. Springer International Publishing, 2019. ISBN: 978-3-030-05716-9.

[RGH+21] Luca Rossetto, Ralph Gasser, Silvan Heller, Mahnaz Parian-Scherb, Loris Sauter, Florian Spiess, Heiko Schuldt, Ladislav Peska, Tomas Soucek, Miroslav Kratochvil, Frantisek Mejzlik, Patrik Vesely, and Jakub Lokoc. On the User-Centric Comparative Remote Evaluation of Interactive Video Search Systems. *IEEE Multimedia*:1–1, 2021. DOI: 10.1109/MMUL.2021.3066779.

[RGL+20] Luca Rossetto, Ralph Gasser, Jakub Lokoc, Werner Bailer, Klaus Schoeffmann, Bernd Muenzer, Tomas Soucek, Phuong Anh Nguyen, Paolo Bolettieri, Andreas Leibetseder, et al. Interactive Video Retrieval in the Age of Deep Learning - Detailed Evaluation of VBS 2019. *IEEE Transactions on Multimedia*. TOMM, 2020.

[RGS+21] Luca Rossetto, Ralph Gasser, Loris Sauter, Abraham Bernstein, and Heiko Schuldt. A System for Interactive Multimedia Retrieval Evaluations. In *Proceedings of the 27th International Conference on Multimedia Modeling*, MMM 2021, pages 385–390, Cham. Springer International Publishing, 2021. ISBN: 978-3-030-67835-7.

[RGG+18] Luca Rossetto, Ivan Giangreco, Ralph Gasser, and Heiko Schuldt. Competitive Video Retrieval with vitrivr. In *Proceedings of the 24th International Conference on Multimedia Modelling*, MMM 2018, pages 403–406, Cham. Springer International Publishing, 2018. ISBN: 978-3-319-73600-6.

[Spi09] David I Spivak. Simplicial Databases. *arXiv preprint arXiv:0904.2012*, 2009.

[WSB98] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In *VLDB*, volume 98, pages 194–205, New York City, NY, USA. Morgan Kaufmann, 1998.

[ZW14] Jan Zahálka and Marcel Worring. Towards interactive, intelligent, and integrated multimedia analytics. In *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 3–12, 2014. DOI: 10.1109/VAST.2014.7042476.

[ZAD+06] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity Search The Metric Space Approach*. Springer Science + Business Media, Inc., 2006. ISBN: 978-0-387-29146-8.

Curriculum Vitae

Name	Ralph Marc Philipp Gasser
	Brunnenweg 10, 4632 Trimbach
Date of Birth	28.03.1987
Birthplace	Riehen BS, Switzerland
Citizenship	Switzerland

Education

since Oct. 2017	Ph. D. in Computer Science under the supervision of Prof. Dr. Heiko Schuldt, Databases and Information Systems Tesearch Group, University of Basel, Switzerland
Sep. 2014 – Jul. 2017	M. Sc. in Computer Science, University of Basel, Switzerland
Sep. 2007 – Jul. 2014	B. Sc. in Nanoscience, University of Basel, Switzerland
Sep. 2002 – Dec. 2006	Matura, Gymnasium Liestal, Liestal, Switzerland

Employment

since Oct. 2017	Research & Teaching Assistant, Databases and Information Systems Research Group, University of Basel, Switzerland
since Dec. 2010	IT Consultant & Partner, pontius software GmbH, Bubendorf, Switzerland
Jun. 2008 - Dec. 2011	Application Manager “imdas pro”, Bildungs-, Kultur- und Sportdirektion des Kantons Basel-Landschaft, Liestal, Switzerland

Publications

2021

- Luca Rossetto, Ralph Gasser, Loris Sauter, et al. A System for Interactive Multimedia Retrieval Evaluations. In *Proceedings of the 27th International Conference on Multimedia Modeling*, MMM 2021, pages 385–390, Cham. Springer International Publishing, 2021. ISBN: 978-3-030-67835-7

- Silvan Heller, Ralph Gasser, Cristina Illi, et al. Towards Explainable Interactive Multi-Modal Video Retrieval with vitrivr. In *Proceedings of the 27th International Conference on Multimedia Modeling*, MMM 2021, pages 435–440, Cham. Springer International Publishing, 2021. ISBN: 978-3-030-67835-7. DOI: 10.1007/978-3-030-67835-7_41

- Luca Rossetto, Ralph Gasser, Silvan Heller, et al. On the User-Centric Comparative Remote Evaluation of Interactive Video Search Systems. *IEEE Multimedia*:1–1, 2021. DOI: 10.1109/MMUL.2021.3066779

2020

- Ralph Gasser, Luca Rossetto, Silvan Heller, et al. Cottontail DB: an Open Source Database System for Multimedia Retrieval and Analysis. In *Proceedings of the 28th ACM International Conference on Multimedia*, ACM MM 2020, pages 4465–4468, 2020 (Best Open Source System Award)

- Samuel Börlin, Ralph Gasser, Florian Spiess, et al. 3D Model Retrieval Using Constructive Solid Geometry in Virtual Reality. In *Proceedings of the 2020 IEEE International Conference on Artificial Intelligence and Virtual Reality*, AIVR 2021, pages 373–374, 2020

- Luca Rossetto, Ralph Gasser, Jakub Lokoc, et al. Interactive Video Retrieval in the Age of Deep Learning - Detailed Evaluation of VBS 2019. *IEEE Transactions on Multimedia*. TOMM, 2020

2019

- Ralph Gasser, Luca Rossetto, and Heiko Schuldt. Multimodal Multimedia Retrieval with vitrivr. In *Proceedings of the 2019 International Conference on Multimedia Retrieval (ICMR 2019)*, ICMR 2019, pages 391–394, New York, NY, USA. Association for Computing Machinery, 2019. ISBN: 978-1-4503-6765-3. DOI: 10.1145/3323873.3326921 (Best Demo Award)

- Jakub Lokoč, Gregor Kovalčík, Bernd Münzer, et al. Interactive Search or Sequential Browsing? a Detailed Analysis of the Video Browser Show-

down 2018. *ACM Transactions on Multimedia Computing, Communications, and Applications*. TOMM, 15(1):1–18, 2019

– Ralph Gasser, Luca Rossetto, and Heiko Schuldt. Towards an all-purpose content-based multimedia information retrieval system. *CoRR*, abs/1902.03878, 2019

– Luca Rossetto, Mahnaz Amiri Parian, Ralph Gasser, et al. Deep learning-based concept detection in vitrivr. In *Proceedings of the 25th International Conference on Multimedia Modelling*, MMM 2019, pages 616–621, Cham. Springer International Publishing, 2019. ISBN: 978-3-030-05716-9

2018

– Luca Rossetto, Ivan Giangreco, Ralph Gasser, et al. Competitive Video Retrieval with vitrivr. In *Proceedings of the 24th International Conference on Multimedia Modelling*, MMM 2018, pages 403–406, Cham. Springer International Publishing, 2018. ISBN: 978-3-319-73600-6

Advised Theses

– Renato Farrugio. *Scene Text Recognition in Images and Video with Cineast*. Bachelor's Thesis, University of Basel

– Florian Mohler. *Online Image Analysis to Identify Politicians in Twitter Images*. Bachelor's Thesis, University of Basel

– Gabriel Zihlmann. *Magnetic Resonance Fingerprinting Reconstruction using Methods from Multimedia Retrieval*. Master's Thesis, University of Basel

– Yan Wang. *Detecting Visual Motives Using Online Clustering of Similar Images*. Bachelor's Thesis, University of Basel

– Samuel Börlin. *3D Model Retrieval using Constructive Solid Geometry in Virtual Reality*. Master's Thesis, University of Basel

– Manuel Hürbin. *Retrieval Optimization in Magnetic Resonance Fingerprinting*. Bachelor's Thesis, University of Basel

– Loris Sauter. *Fighting Misinformation: Image forgery detection in social media streams*. Master's Thesis, University of Basel

Honors and Awards

Best VBS System Award for the winning system of the Video Browser Showdown (VBS) and paper titled *Towards Explainable Interactive Multi-Modal Video Retrieval with vitrivr* at the *27th International Conference on Multimedia Modeling (MMM 2021)*

Best Demo Award for the best demo paper titled *A System for Interactive Multimedia Retrieval Evaluations* at the *27th International Conference on Multimedia Modelling (MMM 2021)*

Best Open Source Award for the best open source system paper titled *Cottontail DB: An Open Source Database System for Multimedia Retrieval and Analysis* at the *28th ACM International Conference on Multimedia (ACM MM 2020)*

Best Demo Award for the best demo paper titled *Multimodal Multimedia Retrieval with vitrivr* at the *ACM International Conference on Multimedia Retrieval (ICMR 2019)*

Best VBS System Award for the winning system of the Video Browser Showdown (VBS) and paper titled *Deep Learning-based Concept Detection in vitrivr* at the *25th International Conference on Multimedia Modeling (MMM 2019)*

Fritz Kutter Award 2017 for the Master's Thesis titled *Towards an All-Purpose, Content-based Multimedia Retrieval System*. The Fritz Kutter trust managed by the ETH Zürich honors the best thesis (diploma, master, doctoral) at a Swiss University every year.

Other Relevant Experience

- Technical Program Committee Member (PC Member) of the *International Conference on Multimedia Modeling (MMM 2022)*
- Technical Program Committee Member (PC Member) of the *International Conference on Multimedia Retrieval (ICMR 2021), Workshop – Lifelog Search Challenge*
- Technical Program Committee Member (PC Member) of ViRaL 2021

Ask Luca what exactly this

– Technical Program Committee Member (PC Member) of the *ACM International Conference on Multimedia (ACM MM 2020)*

Declaration on Scientific Integrity

includes Declaration on Plagiarism and Fraud

Author
Ralph Marc Philipp Gasser

Matriculation Number
2007-050-131

Title of Work
Data Management for Dynamic Multimedia Analytics and Retrieval

PhD Subject
Computer Sciences

Declaration
I hereby declare that this doctoral dissertation *"Data Management for Dynamic Multimedia Analytics and Retrieval"* has been completed only with the assistance mentioned herein and that it has not been submitted for award to any other university nor to any other faculty at the University of Basel.

Basel, DD.MM.YYYY

Signature

Hand-in separately