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

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 [**Zahalka:2014towards**]. 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 [**Lowe:1999object**; **Bay:2006surf**] 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 [**Goodfellow:2016deep**].

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 [**Blanken:2007multimedia**]. On the one hand, given the structure of a relational database and languages like SQL, a user can specify exactly

what elements 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* [**Cao:2010mind**; **Ghias:1995query**; **Boerlin:20203d**].

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 [**Lokovc:2019interactive**]. 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 [**Rossetto:2020interactive**]. 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 [**Chinchor:2010multimedia**; **Keim:2010mastering**]. 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.

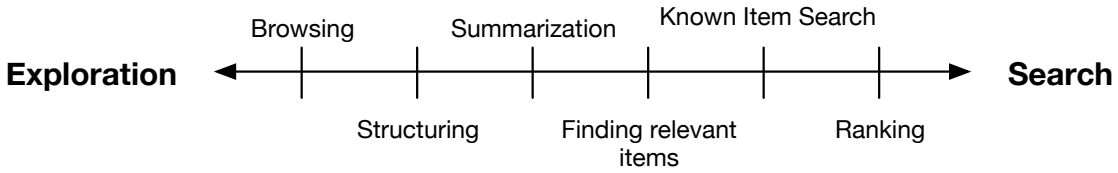

Figure 1.1 Exploration-search axis of multimedia analytics [Zahalka:2014towards].

For analytics on a multimedia collection, Zahalka et al. [**Zahalka:2014towards**] propose the formal model of an *exploration-search axis*, which is depicted in **??**. 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 [**Jonson:2016**]. 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 **??**.

1.2 Research Gap and Objective

It has been pointed out by Jonson et al. [**Jonson:2016**] (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 [**Giangreco:2016adam**; **Giangreco:2018thesis**] and multimedia retrieval systems that refactor data management into distinct components [**Rossetto:2018thesis**], that statement, to some extent, still holds true today. While [**Giangreco:2018thesis**] 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 [**Giangreco:2016adam**; **Giangreco:2018thesis**] and motivated by the "Ten Research Questions for Scalable Multimedia Analytics" [**Jonson:2016**], 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 [**Giangreco:2018thesis**] and [**Rossetto:2018thesis**] 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 [**Petrov:2019Database**]. 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 **[Jonson:2016]**, 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 **??**.

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 deteriorates 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.

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 [**Weber:1998va**], LSH [**Indyk:1998lsh**], PQ [**jegou:2011pq**] 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	

1.3 Contribution

In this thesis, we try to address the research gap identified and described in **??** 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* [**Gasser:2020cottontail**].
- 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 (**??**) as well as an overview of relevant research done in the fields of multimedia retrieval, multimedia analytics and data management (**??**).

The second part, gives a brief summary of the theoretical foundation in multimedia analysis & retrieval (**??**) and databases (**??**) 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 (**??**) and introduces our reference implementation Cottontail DB (**??**).

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

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 Systems (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 DBMS systems were introduced in the 1960s [**Garcia:2009Database**] 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:

- DBMS manage data corpora that can range from a few megabytes to houndreds of terrabytes in size.
- DBMS provide users with the ability to query the data using a data query language (DQL). Such queries answer specific "questions" about the data.
- DBMS provide users with the ability to modify the data using a data manipulation language (DML). Modifications include adding, removing or changing existing entries.
- DBMS usually assure durability upon failure, provide access control for concurrent read/write operation and means to manage data's logical structure.

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. Most of the basic aspects are inspired by [**Garcia:2009Database**] and [**Petrov:2019Database**].

4.1 The Relational Data Model

A data model is a formal framework that describes any type of data or information. It usually involves a description of the data's *structure*, the *operations* that can be performed on the data and the *constraints* that should be imposed [**Garcia:2009Database**]. The purpose of any data model is to formalize how data governed by it can be accessed, modified and queried.

In the context of database systems, it has become common practice to make a clear distinction between the *physical data model* and the a higher-level *logical data model*. While the former captures low-level structure (representation in bits and bytes, data structures etc.) and operations (reads and writes), the latter describes higher-level semantics (attributes, relationships etc.). The argument for this separation is that "users" of any DBMS should not concern themselves with how exactly data is organized and accessed at the lowest level.

In June 1970, E. F. Codd published his pivotal research article *Relational Model of Data for Large Shared Data Banks* [**Codd:1970Relational**] in which he describes such a logical data model for databases, which he himself refers to as "relational". This model has become the fundament on which many if not most of the modern database management systems have been built in the past decades.

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*. **??** features a relation *paintings* and each entry in the table represents a painting and the related attribute values for attributes *title*, *artist* and *painted*. Note, that we already use the mathematical notation introduced in **??**.

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

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

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

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 Relation according to [Codd:1970Relational]

Given $i, j, N \in \mathbb{N}_{>0}$ and a collection of sets $\mathcal{D}_1, \mathcal{D}_2, ...\mathcal{D}_N$, a *relation* $\mathcal{R}$ on these sets is a set of *tuples* $t_i = (a_{i,1}, a_{i,2}...a_{i,n}) \in \mathcal{R}$ such that the attribute values $a_{i,1} \in \mathcal{D}_1, a_{i,2} \in \mathcal{D}_2...a_{i,N} \in \mathcal{D}_N$. Ergo, $\mathcal{R}$ is a subset of the Cartesian product of the relation's *data domains* $\mathcal{D}_j$, that is, $\mathcal{D}_1 \times \mathcal{D}_2... \times \mathcal{D}_N$. The number of data domains N is referred to as the relation's *degree* and we call a relation *N-ary*, e.g., binary for $N = 2$.

The relational model does not dictate what the data domains should be. However, in a relational database system, the data domains $\mathcal{D}_1, \mathcal{D}_2, ...\mathcal{D}_N$ usually correspond to the data types supported by the database, for example, `Int` for integer numbers, `Float` for floating point numbers or `Varchar` for text [1]. To keep things simple, we will use *data domain* synonymous for *data type*. The combination of a data domain with a human readable label is called attribute and we will use subscripts to indicate an attribute's name or index, that is, $\mathcal{A}_{name} = (\mathcal{D}, name)$. In its original form, the relational model assumes the following properties to be true for a relation $\mathcal{R}$ and its attributes [**Codd:1970Relational**]:

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

Ordering of attributes Attribute values $a_{i,j}$ always occur in the same order within the $t_i \in \mathcal{R}$, which corresponds to the order of the attributes $\mathcal{A}_j$. This order can evolve over time but remains constant in a momentary snapshot of $\mathcal{R}$.

Duplicates Relations do not allow for duplicates, i.e., every tuple t_i must be unique in terms of their attributes.

Atomicity The attribute values in a tuple t_i are considered to be atomic, i.e., it is not possible to further decompose them.

[1] Strictly speaking as per definition, 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 [**Spivak:2009Simplicial**]

Given the notion and the properties of a relation $\mathcal{R}$, one can introduce the idea of a *heading* or *schema* $SCH(\mathcal{R})$ and *body* or *extent* $EXT(\mathcal{R})$ of a relation. Which are the set of attributes and tuples respectively that make up $\mathcal{R}$.

$$SCH(\mathcal{R}) = \{\mathcal{A}_1, \mathcal{A}_2, ...\mathcal{A}_N\} \tag{4.1}$$

$$EXT(\mathcal{R}) = \{(a_{1,1}, ..., a_{1,N}), ...(a_{M,1}, ..., a_{M,N})\}, a_{i,j} \in \mathcal{D}_j \tag{4.2}$$

The sum of all data contained in a database system can be regarded as a collection of different relations $\mathcal{R}_k, k \in \mathbb{N}_{\geq 0}$ of assorted degrees N_k and data domains $\mathcal{D}_j \in \mathbb{D}$, with $\mathbb{D}$ being the set of all data domains supported by the system. The schema of a database can then be seen as the sum of all $SCH(\mathcal{R}_k)$. As [**Codd:1970Relational**] points out, relations are subject to change over time, which are changes to either $SCH(\mathcal{R})$ or $EXT(\mathcal{R})$, for example, by tuples being added to (insert) or removed from (delete) a relation, or by altering one or multiple attributes.

4.1.1 Keys and Normal Forms

The notion of a relation introduced in the previous section provides us with the basic tools for data modeling. [**Codd:1970Relational**] proposed a wide range of additional constraints to guarantee proper data definition using the relational model and the following, additional definitions.

Primary key A *primary key* is a combination of attributes $P \subset SCH(\mathcal{R})$ that uniquely identify a tuple $t_i \in \mathcal{R}$. That is, the attributes $SCH(\mathcal{R}) \setminus P$ and their values are functionally determined by P.

Foreign key Foreign keys are attributes $F \subset SCH(\mathcal{R})$ that are not primary keys but reference the primary key of R or some other relation R_k. Foreign keys can be used to model relationships between relations.

An example of relations with primary and foreign key attributes is given in **??**. Primary keys are indicated with a * and foreign keys are underlined.

Example 4.2 Relations with Primary and Foreign Keys

The following tables lists the heading and the body of relations $\mathcal{R}_{paintings}$ and $\mathcal{R}_{artists}$. Note that $\mathcal{R}_{paintings}$ only contains attributes that are functionally dependend on the painting, identified by $\mathcal{A}_{title}$. Information about the artist, e.g.

their date of birth, do not depend on the the title of the paining and thus belong to another relation.

$\mathcal{R}_{paintings}$	$\mathcal{A}^*_{title}$	$\mathcal{A}_{artist}$	$\mathcal{A}_{painted}$
t_1	Mona Lisa	1	1506
t_2	The Starry Night	2	1889
t_3	Las Meninas	3	1665

$\mathcal{R}_{artist}$	$\mathcal{A}^*_{id}$	$\mathcal{A}_{fname}$	$\mathcal{A}_{lname}$	$\mathcal{A}_{birth}$	$\mathcal{A}_{death}$
t_1	1	Leonardo	da Vinci	1452	1519
t_2	2	Vincent	van Gogh	1853	1890
t_3	3	Diego	Velázquez	1599	1660

Note, that we used an artifical primary key for $\mathcal{R}_{artists}$, which is often done in practice.

4.1.2 Relational Algebra

4.1.3 Extensions

4.1.3.1 Extended Projection

4.1.3.2 Ranked Relational Algebra

4.1.3.3 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 **??**, one can extend the set 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."* ([**Giangreco:2018thesis**], 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 [**Giangreco:2018thesis**] has a certain elegance to it, it comes with

limitations that become apparent upon disection of τ's structure. In its postulated form, τ addresses several functions at once:

1. It specifies the distance function and its parameters.

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 and optimize, 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 similarity-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 opreations $\tau^{kNN}_{\delta(\cdot,\cdot),a,q}(\mathcal{R})$ 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:

$$\sigma(\tau^{kNN}_{\delta(\cdot,\cdot),a,q}(\mathcal{R})) \neq \tau^{kNN}_{\delta(\cdot,\cdot),a,q}(\sigma(\mathcal{R})) \tag{4.3}$$

These two queries are very different: On the one hand, the first query filters the results of a kNN-search on $\mathcal{R}$, thus returning $n \leq l$ results, wherein $n = k$ only if σ matches all tuples. The second query 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 $\sigma_{\delta \geq \epsilon}$.

$$\sigma(\tau^{\epsilon NN}_{\delta(\cdot,\cdot),a,q}(\mathcal{R})) = \tau^{\epsilon NN}_{\delta(\cdot,\cdot),a,q}(\sigma(\mathcal{R})) \tag{4.4}$$

Hence, σ and $\tau^{\epsilon NN}_{\delta(\cdot,\cdot),a,q}(\mathcal{R})$ commute and yield equivalent results.

4.2 Queries: From Expression to Execution

4.2.1 Structured Query Language (SQL)

4.2.2 Query Planning

4.2.3 Query Execution

4.3 Storage, Indexes and Caching

4.4 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 [**Zezula:2006similarity**] for similarity search presented in **??**, we propose to extend the notion of proximity-based similarity search to that of a more general *proximity based operation (pOP)* following **??**.

Definition 5.1 Proximity Based Operation (pOP)

Any database query operation on relation $\mathcal{R}$ that relies on a notion of distance $d = \delta(a_i, q)$ between an attribute $a_i \in \mathcal{D}_q$ and some query $q \in \mathcal{D}_q$ given $\mathcal{D}_q \in SCH(\mathcal{R})$ and distance function $\delta : \mathcal{D}_q \times \mathcal{D}_q \rightarrow \mathbb{R}$, is called a *proximity based operation*. 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 **??** simply requires a notion of proximity between some attribute and a query to be obtained by means of 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.

It is straightforward to see, that similarity search falls into the broader category of a pOP, wherein the distance is being used to rank the relation and subsequently limit its cardinality. In addition, pOPs include other operations such as calculating the distance value followed by some filtering or grouping based on the computed value. Hence, we do not limit ourselves to simple search.

5.1.1 Revisiting Distance Computation

Since the choice of the distance function δ is a crucial part of any pOP, it is worth revisiting its definition. Again, starting with the metric space modell [**Zezula:2006similarity**], we identify the following constraints with respect to the distance function:

1. 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.
2. The domain (i.e., the input) of the distance function δ is assumed to be $\mathbb{R}^{dim} \times \mathbb{R}^{dim}$, hence, restricted to real-valued vectors and the distance function is assumed to be a binary function.
3. $(\mathcal{D}_q, \delta)$ are supposed to constitute a metric, thus satisfying the non-negativity, identity of indiscernibles, symmetry and subadditivity property.

Upon closer examination, one can see 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 a programming 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 pOPs. If, however, we turn to **??**, we realise that it is not reasonable to restrict the domain of the distance function to R^{dim}

Example 5.1 Maximum Inner Product Search (MIPS) for MRF

In MRF (see **??**), we try to obtain the signal vector $a_{i \in N} \in \mathcal{D}_q \subset \mathbb{C}^{dim}$ so that it maximizes the inner product to a query vector $q \in \mathbb{C}^{dim}$. In this case, the distance function δ has the form $\delta \colon \mathbb{C}^{dim} \times \mathbb{C}^{dim} \rightarrow \mathbb{R}_{\geq 0}$ with $dim \in \mathbb{N}_{>1}$. Hence, to domain of δ is complex.

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 [**Giangreco:2018thesis**]. Still we have to acknowledge, that the text-book definition of a distance function is too limited for many real-world applications, and that the query and probing arguments of a distance function could be any type of value supported by the database management system. For example, the *Levenshtein distance* [**Levensthtein:1965Binary**], often used in computer linguistics, is a distance metric on string values.

If now in addition, we consider **??**, we see that restricting oneself to binary functions is also too limiting when considering 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:

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

The distance function then 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 pOPs must always constitute a metric on some real-valued vector space may be very common [**Zezula:2006similarity**], we see in practice that this assumption is often violated [**Bernhauer:2019Nonmetric**] 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 metrics, it is obvious that such functions exist and must thus be considered in any generalized database application supporting pOPs.

More examples

In order to address the aforementioned limitations while still accomodating classical, metric distance functions, we propose the extension of a the distance function to the notion more general notion of a *Distance Function Class (DFC)* following **??**.

Definition 5.2 Distance Function Classes (DFC)

A *Distance Function Class (DFC)* $\hat{\delta}\colon \mathcal{D}_q \times \mathcal{D}_q \times \mathcal{D}_1 ... \times \mathcal{D}_n \to \mathbb{R}$ is an n-ary but at least binary function $\hat{\delta}(a, q, s_1, ..., s_n)$ that outputs a distance $d \in \mathbb{R}$ between a probing argument $a \in \mathcal{D}_q \subset \mathcal{R}$ 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 $\mathbb{D}$.

As convention for notation, the probing argument a of a DFC $\hat{\delta}(a, q, s_1, ..., s_n)$ will always come first, followed by the query argument q, followed by its support arguments s_k in no particular order.

From the perspective of the database system, DFCs represent a family of higher-order functions available for distance calculation in pOPs. With the idea of a DFC in mind, we can start to reason about their properties in the broader context of database operations in general and pOPs in particular:

DFC and pOPs In extension to **??**, we see that any database query that involves the execution of a DFC falls into the category of a pOP.

Purity Since the purpose is a DFC is to quantify proximity between a probing argument a and a query argument q, we require that $a, q \in \mathcal{D}_q$, i.e., both must be member of the same data domain $\mathcal{D}_q$.

Definition A N-ary DFC is uniquely defined by its *signature*, which is a $(N+1)$-tuple specifying the name of the DFC and the data domains D_i of the arguments it can accept as defined in **??**.

Implementation In the context a concrete pOP instance, the query argument q and the support arguments s_i remain constant. Hence, irrespective of the arity of the DFC, the *implementation* δ of DFC $\hat{\delta}$ is always a unary function with signature $(\texttt{NAME}, \mathcal{D}_q)$, i.e., **??** holds.

Definition 5.3 Signature and Implementation of DFCs

The *signature* of a DFC $\hat{\delta}$ is defined as:

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

The *implementation* δ of a DFC $\hat{\delta}$ is defined as:

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

Very generally, DFCs $\hat{\delta}$ in the context of a database system can be viewed as higher-level functions that parametrize a unary function using the query and support arguments. A very simple and widely used example would be the Manhattan (L1) and the Euclidean (L2) distance, which are both parametrized versions of the more general Minkowski distance $\hat{\delta}_M : \mathbb{R}^{dim} \times \mathbb{R}^{dim} \times \mathbb{R} \rightarrow \mathbb{R}_{\geq 0}$:

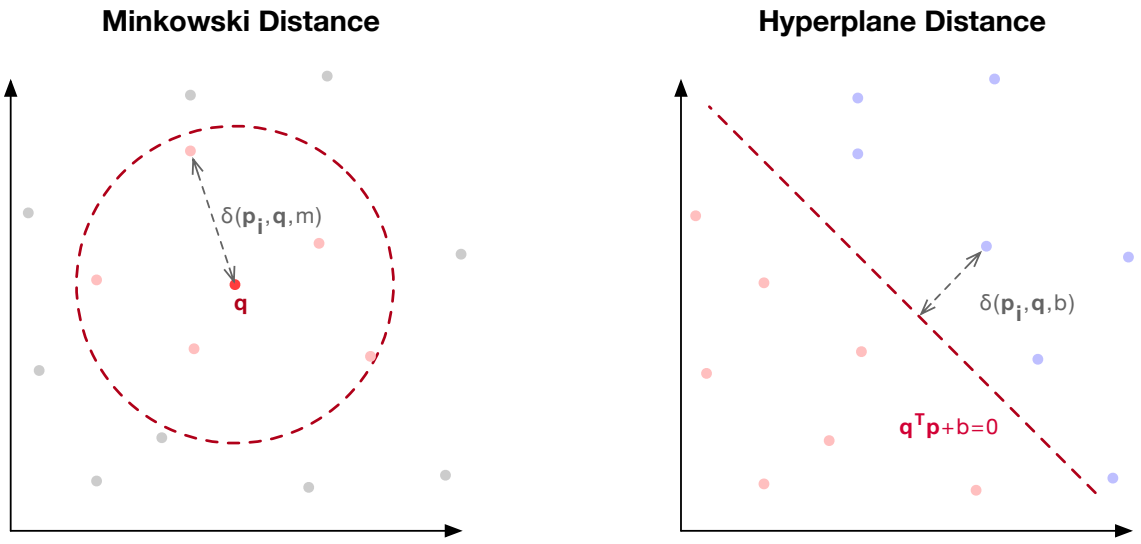

Figure 5.1 Two examples of distance functions that fall into the broader category of a DFC.

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

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

$$\delta_{L2}(\mathbf{a}, \mathbf{q}) = \texttt{IMP}_{p=2}(\hat{\delta}_M) = \sqrt{\sum_{i=1}^{n} \mid q_i - a_i \mid^2} \quad (5.4)$$

We will show in **??**, that a database system can use the *definition* property to plan the execution of a query involving a DFC using optimized versions for different data types and/or substituting DFCs using an index. Furthermore, the *implementation* property can be leveraged to facilitate pre-computation and caching.

Using the idea of a DFC, we can express many different types of distance functions in a simple yet expressive framework. Two examples are illustrated in **??** and it is easy to see that both the functions used in **??** and **??** fall into the category of a DFC.

5.1.1.1 Parametrized dimensionality

As an aside, we must address the role of dimensionality in the case of vector spaces such as $\mathbb{R}^{dim}$ or $\mathbb{C}^{dim}$. One could argue, that the dimensionality of such a vector space can also be seens as a parameter of the SPF. Nevertheless, we consider the dimensionality to be a structural property of the underlying data domain as, for example, the data type. This means, that dimensionality as well

as the type are well-defined and most importantly constant properties for a given relation.

5.1.2 Extending the Relational Model

Using the definition of a DFC as described in **??**, one can start to integrate these into the relational algebra model. Following [**Giangreco:2018thesis**], we first assume the set of supported data domains $\mathbb{D}$ to be extended by whatever data domain $\mathcal{D}$ is desired. Following that recipe, a database management system can be extended to support every type of (mathematical) object that is useful for a given application.

Using the idea of an extended projection $\pi_{\mathcal{A}}$ as defined in **??** and described, for example, by [**Garcia:2009Database**], the invocation of a DFC can be expressed as follows.

Definition 5.4 Distance Function Classes (DFC) in Extended Projection

Let $\delta\colon \mathcal{D}_q \times \mathcal{D}_q \times \mathcal{D}_1 ... \times \mathcal{D}_{n-2} \rightarrow \mathbb{R}$ be a DFC and $\mathcal{R}$ be a relation with $SCH(\mathcal{R}) = \{a_1, a_2, ...a_n\}$. The *extended projection* $\pi_{\delta(a_1,a_2,...,a_n)}(\mathcal{R})$ describes the execution of the n-ary DFC δ using attributes $a_1, a_2, ...a_n$ from relation $\mathcal{R}$ as parameters. Note that $\pi_{\delta(a_1,a_2,...,a_n)}$ introduces a new, calculated distance attribute $a_d \in \mathbb{R}$ on each tuple $t_i \in \mathcal{R}$, i.e., $SCH(\pi_{\delta(a_1,a_2,...,a_n),a_1,a_2,...,a_n}) = SCH(\mathcal{R}) \cup \{a_d\}$.

Obviously, the combination of multiple DFCs in a single, extended projection or the combination of simple attribute projection with DFCs are also allowed. Hence, the following expressions are valid examples of the extended projection on relation $\mathcal{R}$ with $SCH(\mathcal{R}) = \{a_1, a_2, a_3, a_4\}$: $\pi_{\delta_1(a_1,a_2),\delta_2(a_2,a_3,a_1)}(R)$ or $\pi_{\delta(a_1,a_2),a_3,a_4}(\mathcal{R})$ or $\pi_{a_1,a_3,a_4}(\mathcal{R})$.

- NNS: Scan -> (Predicate) -> Distance Function -> Sort -> Limit -> (Predicate); no need for dedicated language feature aside from distance function
- Distance function is a binary function $D(q, v) \longrightarrow d$, q and v can be elements of $\mathbb{R}^d, \mathbb{C}^d$ or some other object (e.g. matrices)
- Different types of distance functions, depending on parameters they accept; e.g. distance between point & point or point & plane etc.
- Systems perspective 1: How enable planner to reason about distance function execution? Possible optimizations?

– Systems perspective 2: Concrete applications in multimedia retrieval and analytics?

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 5.2 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 **??**

– 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

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 Jan. 2000	Ph. D. in Computer Science under the supervision of Prof. Dr. Heiko Schuldt, Databases and Information Systems research group, University of Basel, Switzerland
Sept. 1997 – Aug. 1999	M. Sc. in Computer Science, University of Basel, Switzerland
Sept. 1994 – Aug. 1997	B. Sc. in Computer Science, University of Basel, Switzerland

Employment

since Jan. 2000	Research and teaching assistant, Databases and Information Systems research group, University of Basel, Switzerland

Publications

1937

- Test

Hand-in in thesis and separately

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