Quantum Query Processing: Unifying Database Querying and Information Retrieval

Ingo Schmitt

Otto-von-Guericke-Universität Magdeburg
Institut für Technische Informationssysteme
Postfach 4120, D–39016 Magdeburg, Germany
E-mail: schmitt@iti.cs.uni-magdeburg.de

September 2006

Abstract: Traditional database query languages are based on set theory and crisp first order logic. However, many applications require retrieval-like queries which return result objects associated with a degree value of being relevant to the query. Traditionally, retrieval systems estimate relevance by exploiting hidden object semantics whereas query processing in database systems relies on matching select-conditions with attribute values. Thus, different mechanisms were developed for database and information retrieval systems. In consequence, there is a lack of support for queries involving both retrieval and database search terms. In this work, we develop a unifying framework based on the mathematical formalism of quantum mechanics and quantum logic. Van Rijsbergen already discussed the strong relation between the formalism of quantum mechanics and information retrieval. The goal of this work is to interrelate concepts from database query processing to concepts from quantum mechanics and logic. As result, we obtain a common theory which allows us to incorporate seamlessly retrieval search into traditional database query processing. Exploiting our theoretical results, we introduce the quantum query language QQL. In contrast to competing approaches, our formalism is based on quantum logic.

Contents

Abstract: Traditional database query languages are based on set theory and crisp first order logic. However, many applications require retrieval-like queries which return result objects associated with a degree value of being relevant to the query. Traditionally, retrieval systems estimate relevance by exploiting hidden object semantics whereas query processing in database systems relies on matching select-conditions with attribute values. Thus, different mechanisms were developed for database and information retrieval systems. In consequence, there is a lack of support for queries involving both retrieval and database search terms. In this work, we develop a unifying framework based on the mathematical formalism of quantum mechanics and quantum logic. Van Rijsbergen already discussed the strong relation between the formalism of quantum mechanics and information retrieval. The goal of this work is to interrelate concepts from database query processing to concepts from quantum mechanics and logic. As result, we obtain a common theory which allows us to incorporate seamlessly retrieval search into traditional database query processing. Exploiting our theoretical results, we introduce the quantum query language QQL. In contrast to competing approaches, our formalism is based on quantum logic.

Chapter 1

Introduction

In several application areas, e.g. in multimedia or in XML-applications, expressing an information need often requires a mixture of traditional database [Cod71, Mai83, DD97], retrieval-like [BR99, vR79], and proximity search terms. Retrieval functionality, for instance, is required if database objects are to be searched by a notion of similarity. For example, consider a collection of XML-documents about paintings. Each contains a textual content description within the <`desc`> tag, the paint technique within the <`technique`> tag, and the century of its creation within the <`century`> tag. The query *'retrieve all oil paintings showing evening twilight painted about in 16th century'* combines conjunctively a database query (`technique='oil'`), a text retrieval query (`desc is about 'evening twilight'`), and a proximity query (`century` $\approx$ `16th`).

Information retrieval systems return every result object equipped with a so-called similarity score which is usually understood as estimated degree of the corresponding object being relevant to the query. Another type of queries are *proximity queries* being insufficiently supported by traditional database systems. It introduces the notion of proximity among values producing non-discrete truth values.

Traditional database query languages like the relational calculus offer Boolean algebra operators to construct complex search conditions from atomic conditions. However, such operators deal with Boolean truth values only. Furthermore, deciding whether a database object belongs to a query result is based on simple comparisons with attribute values. In retrieval systems, however, the information being required for *exactly* evaluating a retrieval object against a query is not explicitly available. Thus, a retrieval system can only estimate an object's relevance to a query.

Historically, for database querying and information retrieval different mechanisms have been developed causing a problem for complex queries which involve a combination of retrieval and database query elements. So far, no satisfactory common formalism exists to process such kind of queries.

We state following requirements for a query language basing on a unifying framework:

1. *database query support:* The language must be relational complete.

2. *information retrieval support:* The language must enable us to formulate and to evaluate retrieval-like and proximity query terms.

3. *unifying theoretical framework:* For the language there must exist *one* unifying theoretical framework.

A language which meets these requirements is therefore an extension of a classical database query language. Furthermore, it supports a conjunctive or disjunctive combination of query terms of different types (database, retrieval, proximity).

In this work, we explain how *quantum mechanics* and *quantum logic* provide us a unifying framework for querying databases and retrieval systems. Quantum mechanics comes with its own mathematical formalism. This formalism is attractive for tackling our problem since it combines in a very elegant way concepts from geometry (linear algebra and Hilbert space), logic (quantum logic as a non-standard logic), and probability theory (Gleason's theorem). Van Rijsbergen [vR04] already discussed the strong relation between the formalism of quantum mechanics and information retrieval concepts. Our focus, however, is on mapping concepts from database query processing to the formalism of quantum processing and on establishing, hereby, a connection to information retrieval. The goal of our work is to establish a unifying framework and to develop the quantum query language QQL. We incorporate the notion of similarity and proximity into database query processing by applying the formalism of quantum mechanics and quantum logic. Database tuples are represented by vectors whereas a query corresponds to a vector subspace. Query evaluation is based on computing the squared cosine of the minimal angle between them. We will show that although our framework is based on complex linear algebra concepts query evaluation can be performed using simple arithmetics.

After discussing related work in Chapter 2, we introduce basic concepts of quantum mechanics in Chapter 3 and quantum logic in Chapter 4. The main chapter is Chapter 5 where we demonstrate how traditional database queries basing on Boolean logic are mapped to quantum theory. Furthermore, in this framework, we extend the power of database queries to cope seamlessly with proximity and retrieval search terms.

Chapter 2

Related Work

First order logic is a main concept of database query languages like relational calculus, SQL, and XQuery. Unfortunately, that logic is not adequate for processing queries which combine retrieval and traditional database search conditions. For example, the query `technique = 'oil'` as a typical database query returns a set of paintings for which that condition holds. Contrarily, the query `desc is about 'evening twilight'` is a text retrieval query returning a list of paintings sorted in descending order by their respective similarity scores. Assume, we conjunctively combine both queries into one query:

```
technique = 'oil' AND desc is about 'evening twilight'
```

What would be the result, a list or a set of images? The problem here is the illegal logical combination of an exact query providing us Boolean values with an imprecise retrieval query returning similarity scores from the interval $[0, 1]$. There are two prominent approaches to deal with that conflict.

Boolean Query: The idea realized in most Boolean-logic-based query systems like in the commercial database system DB2 (text extender) is to transform any retrieval query into a Boolean one. This is simply achieved by applying a threshold value. That is, every similarity score greater than the threshold is considered true otherwise false. There are several drawbacks. First, finding a suitable threshold value is not an easy task. Second, as result, we lose information of what degree the similarity condition holds. Thus, we cannot discriminate among paintings from the result set w.r.t. their similarity to evening twilight. Especially in queries composed of several conditions we need that similarity scores.

Retrieval Query: The idea here is to transform the database query into a kind of a retrieval query. That is, logic values from the database query evaluation are mapped to the score values 1 for `true` and to 0 otherwise. These scores can now be arithmetically combined with scores from a retrieval query, e.g. by a simple weighted sum. However, it is not clear at all which aggregation formula should be applied for a specific query. There is a plethora of possible aggregation formulas for that scenario. Furthermore, there is no logic framework (conjunction, disjunction, negation) supporting the formulation of complex queries.

Summarizing, the first approach lacks support for similarity scores whereas the second one fails with respect to an available logic for query formulation and processing.

approach	scores	distributivity	non-dominating
Boolean query	no	yes	–
retrieval query	yes	–	–
fuzzy logic (`min/max`)	yes	yes	no
fuzzy logic (not `min/max`)	yes	no	yes
weighted fuzzy logic	yes	no	–
quantum query language	yes	yes	yes

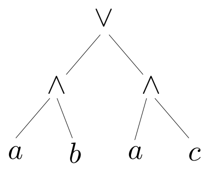
Table 2.1: Properties of different approaches to combine retrieval and database queries

A straightforward solution to the problem is to take advantage of fuzzy logic [Fag98] as proposed, for example, in [Zad88]. In fuzzy logic, similarity scores as well as Boolean truth values are interpreted as fuzzy set membership values which can be combined via logical junctors following complex. Scoring functions t-norm and t-conorm generalize the logical conjunction and disjunction, respectively. Examples of query languages based on fuzzy logic are the `same` algebra [CMPT00], WS-QBE, $\mathcal{SDC}$, and $\mathcal{SA}$ as proposed in [SS04, SSH05]. Fagin's weighting schema [Fag98] is used in those languages in order to equip search conditions with different weights of importance. Bellmann and Giertz [BG73] proved that fuzzy logic with t-norm `min` for conjunction and t-conorm `max` for disjunction obeys the rules of the Boolean algebra. Thus, most query processing techniques known from the database theory are still valid.

The idea of using fuzzy logic for database management is not new. At the beginning of the nineties, techniques of fuzzy logic [Zad88] were applied to traditional database technology in order to cope with vagueness. An overview is given in [GUP05]. Much research was done on developing fuzzy-databases with corresponding fuzzy query languages. [Bol94] introduces a fuzzy ER-model together with a calculus language using fuzzy-logic. Other examples are [GMPC98, BP95] which investigate how to develop a fuzzy-SQL language. [Tak93] sketches the design of a fuzzy calculus, fuzzy algebra, and a mapping between them. However, this work suffers from an incomplete formalization.

Nevertheless, there are some common problems of the fuzzy approaches in our context. First, applying the standard fuzzy norms `min` and `max` suffers from a specific property: The minimum as well as the maximum of two certain scores returns always just the smaller (greater) one of them and ignores completely the greater (smaller) one. For example, assume two conjunctively combined retrieval conditions. The condition which returns smaller scores dominates the result semantics. Contrarily, a *non-dominating* t-norm which respects both scores simultaneously would better meet our understanding of query combination. Actually, fuzzy logic introduces different non-dominating t-norms, e.g. the algebraic product. Unfortunately, none of them holds idempotence. Thus, in combination with a t-conorm, e.g. the algebraic sum, distributivity cannot be guaranteed. Furthermore, we are faced with problems of failing associativity and distributivity [SS02] when Fagin's weighting schema is applied to a t-norm-conorm-pair. Table 2.1 summarizes the properties of the approaches discussed so far. Later we will show that our proposed language QQL (quantum query language) fullfills all three properties.

The problem of dominance turns out to be even more serious when we examine the way how fuzzy logic is utilized for query evaluation. As shown in Fig. 2.2 (left), fuzzy-based query processing relies on *importing* scores and truth values and interpreting them as membership



e1: $\max(\min(a,b),\min(a,c))$
e2: $a*b+a*c-a*b*a*c$
e3: $a(b+c-bc)$

a	**b**	**c**	**e1**	**e2**	**e3**
0.5	0.5	0	**0.5**	0.25	0.25
0.5	0.5	1	0.5	**0.625**	0.5

Figure 2.1: Example query and three evaluations $e1, e2$, and $e3$

values. Thus, the *generation* of membership values is *not* under control of fuzzy logic. In consequence, there is a high risk that scores are incommensurable due to different scoring functions. Thus they produce an error-prone dominance. Figure 2.3 depicts exemplarily two incommensurable fictive scoring functions. Such a monotonic increasing scoring function maps perceived similarity values to jugded similarity values (scores). Due to the high number of different ways to calculate similarity scores, including e.g. distance-based and cosine-based, incommensurability is very likely to occur. If we combine the scores from both function in Figure 2.3 by using the `min`-function the scores from function A would dominate the ones from function B. Even worse, assume `a` and `b` are differently perceived similarity values w.r.t. two properties. Using incommensurable scoring functions can even swap the order of scores (`b < a` but `a' < b'`) making any comparison meaningless.

Example 2.1 The example query given in Figure 2.1 left demonstrates the problem of dominance and simply importing scores into a fuzzy-based formalism. Evaluation e1 is the standard t-norm/conorm evaluation and e2 uses the algebraic product and sum. For the given a, b, c values, both evaluations produce unexpected overall scores (if interpreted as probability values) which are highlighted as bold numbers. The problems are caused by dominance (e1) and missing distributivity (e2). In consequence, we conclude that a correct dealing with distributivity requires more than just importing the values for a, b, and c into a fuzzy formalism. Instead, we apply a quantum-based approach. The correct evaluation formula presented as e3 is obtained by applying our quantum-based evaluation algorithm.

A different approach to combine the worlds of information retrieval and database proposed in [FR97] is to apply probability theory directly. There, the relational model and the relational algebra are enhanced by the concept of probability. To every tuple an event expression is assigned which allows the computation of probability values. Basic events are assumed to be assigned to explicitly given probability values. Here again, score values are simply imported in the formalism.

Our idea is to take quantum mechanics and logic as formalism to unify the generation of similarity and proximity scores, classical database evaluations as well as their combination via a logic, see Figure 2.2 (right). In this way, we alleviate the problem of incommensurability[1].

The development of quantum mechanics dates back to the beginning of the last century. The theory was strongly influenced by famous physicists like Einstein, Planck, Bohr, Schrödinger,

[1]In fact, many researchers believe that the problem of incommensurability cannot be completely solved.

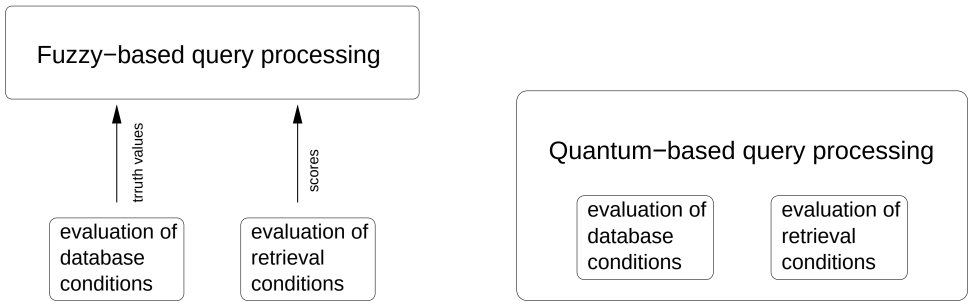

Fuzzy−based query processing
truth values
scores
evaluation of database conditions
evaluation of retrieval conditions
Quantum−based query processing
evaluation of database conditions
evaluation of retrieval conditions

Figure 2.2: Fuzzy evaluation by importing truth and score values (left) and quantum evaluation (right)

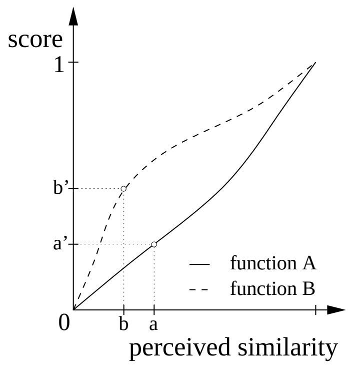


Figure 2.3: Score values from two different scoring functions

and Heisenberg. It deals with specific phenomena of elementary particles like uncertainty of measurements in closed microscopic physical systems and entangled states. In last years, quantum mechanics became an interesting topic for computer scientists who try to exploit its power to solve computationally hard problems. The works [Gru99, CNC00, RP00] provide non-physicists an introduction to quantum computing.

One appealing part of the mathematical formalism of quantum mechanics is quantum logic initially developed by von Neumann [vN32]. Quantum logic, see [BvF81, Loc85a, Loc85b, Zie05], is a non-standard logic based on projectors of a complex separable Hilbert space.

Many concepts of information retrieval [BR99, vR79] are embedded in the formalism of linear algebra and probability theory. Van Rijsbergen as one prominent information retrieval expert discusses in his book [vR04] the strong relationship between concepts of quantum mechanics and information retrieval. We establish here the relationship to database query processing.

Chapter 3

Quantum Mechanics and its Relation to Probability Theory

This chapter gives a short introduction to the formalism of quantum mechanics and its relation to probability theory. After introducing some notational conventions, we briefly present the four postulates of quantum mechanics. Here, we assume the reader being familiar with linear algebra.

The formalism of quantum mechanics deals with vectors of a complex separable Hilbert space **H**. Without losing generality in our context and for a better understanding we restrict our formalism to the real-valued vector space $\mathbb{R}^n$ equipped with the standard scalar product as inner product. The Dirac notation [Dir58] provides an elegant means to formulate basic concepts of quantum mechanics:

- A so-called *ket* vector $|x\rangle$ represents a column vector identified by `x`. Let two special predefined ket vectors be $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.
- The transpose of a ket $|x\rangle$ is a row vector $\langle x|$ called *bra* whereas the transpose of a bra is again a ket. Both form together a one-to-one relationship.
- The *inner product* between two kets $|x\rangle$ and $|y\rangle$ returning a scalar equals the scalar product defined as the product of $\langle x|$ and $|y\rangle$. It is denoted by a *bracket* '$\langle x|y\rangle$'. The *norm* of a ket vector $|x\rangle$ is defined by $||\,|x\rangle\,|| \equiv \sqrt{\langle x|x\rangle}$.
- The *outer product* between two kets $|x\rangle$ and $|y\rangle$ is the product of $|x\rangle$ and $\langle y|$ and is denoted by '$|x\rangle\langle y|$'. It generates a linear operator expressed by a matrix.
- The *tensor product* between two kets $|x\rangle$ and $|y\rangle$ is denoted by $|x\rangle \otimes |y\rangle$ or short by $|xy\rangle$. If $|x\rangle$ is `m`-dimensional and $|y\rangle$ `n`-dimensional then $|xy\rangle$ is an `m`·`n`-dimensional ket vector. The tensor product of two-dimensional kets $|x\rangle$ and $|y\rangle$ is defined by:

$$|xy\rangle \equiv |x\rangle \otimes |y\rangle \equiv \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \otimes \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \equiv \begin{pmatrix} x_1 y_1 \\ x_1 y_2 \\ x_2 y_1 \\ x_2 y_2 \end{pmatrix}.$$

The tensor product between matrices is analogously defined.

Next, we sketch the famous four postulates of quantum mechanics:

Postulate 1: Every closed physical microscopic system corresponds to a separable complex Hilbert space[1] and every state of the system is completely described by a normalized (the norm equals one) ket vector $|\varphi\rangle$ of that space.

Postulate 2: Every evolution of a state $|\varphi\rangle$ can be represented by the product of $|\varphi\rangle$ and an orthonormal[2] operator O. The new state $|\varphi'\rangle$ is given by $|\varphi'\rangle = O|\varphi\rangle$. It can be easily shown that an orthonormal operator cannot change the norm of a state: $||\, O|\varphi\rangle\, || = ||\, |\varphi\rangle\, || = 1$.

Postulate 3: This postulate describes the measurement of a state which means to compute the probabilities of different outcomes. If a certain outcome is measured then the system is automatically changed to that state. Here, we focus on a simplified measurement given by projectors (each one represents one possible outcome and is bijectively associated with one vector subspace). A *projector* $p = \sum_i |i\rangle\langle i|$ is a symmetric ($p^t = p$) and idempotent ($pp = p$) operator defined over a set of orthonormal vectors $|i\rangle$. Multiplying a projector with a state vector means to project the vector onto the respective vector subspace. The probability of an outcome corresponding to a projector p and a given state $|\varphi\rangle$ is defined by

$$\langle\varphi|p|\varphi\rangle \quad = \quad \langle\varphi| \left(\sum_i |i\rangle\langle i| \right) |\varphi\rangle = \sum_i \langle\varphi|i\rangle\langle i|\varphi\rangle.$$

Thus, the probability value equals the squared length of the state vector $|\varphi\rangle$ after its projection onto the subspace spanned by the vectors $|i\rangle$. Due to normalization, the probability value, furthermore, equals geometrically the squared cosine of the minimal angle between $|\varphi\rangle$ and the subspace represented by p.

Postulate 4: This postulate defines how to assemble various quantum systems to one system. The base vectors of the composed system are constructed by applying the tensor product '$\otimes$' on the subsystem base vectors.

Figure 3.1 illustrates the connection between quantum mechanics and probability theory for the two-dimensional case. Please notice that the base vectors $|0\rangle$ and $|1\rangle$ are orthonormal. The measurement of the state $|\varphi\rangle = a|0\rangle + b|1\rangle$ with $||\, |\varphi\rangle\, || = 1$ by applying the projector $|0\rangle\langle 0|$ provides the squared portion of $|\varphi\rangle$ on the base vector $|0\rangle$ which equals a^2. Analogously, the projector $|1\rangle\langle 1|$ provides b^2. Due to Pythagoras and the normalization of $|\varphi\rangle$ both values sum up to one. In quantum mechanics where $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$ represent two possible outcomes of a measurement the values a^2 and b^2 give the probabilities of the respective outcomes. Note that both outcomes correspond to independent events ($\langle 0|1\rangle = 0$) and together they cover the complete event space ($|0\rangle\langle 0| + |1\rangle\langle 1| = I$).

Following [vR04] we discuss two aspects which explain why quantum mechanics may serve as an appropriate model for information retrieval:

[1]For simplicity, we restrict ourselves to the vector space $\mathbb{R}^n$.

[2]An operator O is orthonormal if and only if $O^tO = OO^t = I$ holds where the symbol '^{t}' denotes the transpose of a matrix and 'I' denotes the identity matrix.

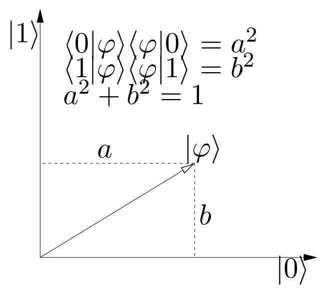

$|1\rangle$
$\langle 0|\varphi\rangle\langle\varphi|0\rangle = a^2$
$\langle 1|\varphi\rangle\langle\varphi|1\rangle = b^2$
$a^2 + b^2 = 1$
a
$|\varphi\rangle$
b
$|0\rangle$

Figure 3.1: Pythagoras and probabilities

1. Information retrieval means to estimate the probability that a database object is relevant to a given retrieval query. One frequently used retrieval model is the vector space model where the scalar product is utilized for the required estimation. Thus, quantum measurement is conceptually very near[3] to the vector space model.

2. Quantum mechanics provides an elegant framework for unifying the notion of geometry, probability, and logic[4]. Gleason's theorem [Gle57] establishes the connection between probability theory and geometry. First, we define a *probability measure*. Let the join (disjunction) of projectors be denoted by '$\vee$', $\mathbf{H}$ be a vector space, $L(\mathbf{H})$ be the set of all subspaces of $\mathbf{H}$, and p be a projector which bijectively corresponds to a subspace from $L(\mathbf{H})$.

   Definition 3.1 (probability measure) *A (countably additive) probability measure on $L(\boldsymbol{H})$ is a mapping $\mu : L(\boldsymbol{H}) \rightarrow [0,1]$ such that $\mu(I) = 1$ and, for any sequence of pair-wise orthogonal[5] projectors p_i and $i = 1, 2, \ldots$: $\mu(\vee_i p_i) = \sum_i \mu(p_i)$.*

   Theorem 3.1 (Gleason) *Let $\boldsymbol{H}$ be a vector space having more than two dimensions. Then every countably additive probability measure on $L(\boldsymbol{H})$ has the form $\mu(p) = \langle\varphi|p|\varphi\rangle$ for a normalized vector $|\varphi\rangle$ of $\boldsymbol{H}$.*

The main idea of supporting database querying by the formalism of quantum mechanics is to model database objects as state vectors and queries as projectors within a well-designed vector space. A state vector as database object encapsulates all the possible results of potential measurements whereas projectors as queries define subspaces. Together they form a probability measure. Table 3.1 relates concepts from database querying to concepts from quantum mechanics.

[3]Quantum measurement yields the squared cosine whereas the vector space model returns the cosine of the enclosed angle.
[4]Quantum logic will be introduced in next chapter .
[5]Orthogonality between two projectors is symmetric and defined by $p_1 p_2 = p_2 p_1 = 0$.

database querying	**quantum mechanics**
database tuple	state vector
query	projector
query processing	quantum measurement
truth values	probability values
boolean logic	quantum logic

Table 3.1: Related concepts from database querying and quantum mechanics.

Chapter 4

Quantum Logic

Following [Zie05], we develop here the main concepts of quantum logic originally developed by von Neumann [vN32]. The starting point is the set P of all projectors of a vector space $\mathbf{H}$ of dimensions greater than two. Each projector $p \in P$ is bijectively related to a closed subspace via $p(\mathbf{H}) = \{p|\varphi\rangle \mid |\varphi\rangle \in \mathbf{H}\}$. The subset relation $p_1(\mathbf{H}) \subseteq p_2(\mathbf{H})$ on P which is equivalent to $p_2p_1 = p_1p_2 = p_1$ forms a complete poset. Furthermore, we obtain a lattice[1] with the binary operations `meet` ($\wedge$) and `join` ($\vee$) being defined as

$$\begin{aligned} p_{p_1(\mathbf{H})} \wedge p_{p_2(\mathbf{H})} &\equiv p_{p_1(\mathbf{H}) \cap p_2(\mathbf{H})} \\ p_{p_1(\mathbf{H})} \vee p_{p_2(\mathbf{H})} &\equiv p_{closure(p_1(\mathbf{H}) \cup p_2(\mathbf{H}))}. \end{aligned}$$

Quantum logic in general does not constitute a Boolean logic since the distribution law is violated. For example, if $|x\rangle$ and $|y\rangle$ were two mutually orthonormal ket vectors then we can define three projectors: $p_1 = |x\rangle\langle x|$, $p_2 = |y\rangle\langle y|$, and $p_3 = |v\rangle\langle v|$ where $|v\rangle = (|x\rangle + |y\rangle)/\sqrt{2}$. Then we obtain

$$p_3 \wedge (p_1 \vee p_2) = p_3 \neq 0 = 0 \vee 0 = (p_3 \wedge p_1) \vee (p_3 \wedge p_2)$$

violating the distribution law. The negation (orthocomplement) for our quantum logic is defined as $\neg p \equiv I - p$ encompassing all projectors orthogonal to p. Including the negation, we obtain an ortholattice fulfilling (1) *compatibility* ($p_1(\mathbf{H}) \subseteq p_2(\mathbf{H}) \implies \neg p_2(\mathbf{H}) \subseteq \neg p_1(\mathbf{H})$) and (2) *invertibility* ($p \vee \neg p = I, p \wedge \neg p = 0,$ and $\neg\neg p = p$). From these laws the de Morgan laws can be derived. The ortholattice of projectors fulfills furthermore the law of orthomodularity ($p_1(\mathbf{H}) \subseteq p_2(\mathbf{H}) \implies p_1 \vee (\neg p_1 \wedge p_2) = p_2$) providing us an orthomodular lattice of projectors.

In this work, we have to embed Boolean logic exploited from relational calculus into quantum logic. Actually, quantum logic can be seen as a generalization of a Boolean logic: The sublattice over every equivalence class comprising *commuting* projectors constitutes a boolean logic.

Definition 4.1 (commuting projectors) *Two projectors p_1 and p_2 of a vector space $\boldsymbol{H}$ are called* commuting projectors *if and only if $p_1p_2 = p_2p_1$ holds.*

From linear algebra we know that two projectors $p_1 = \sum_i |i\rangle\langle i|$ and $p_2 = \sum_j |j\rangle\langle j|$ commute if and only if their ket vectors $|i\rangle$ and $|j\rangle$ are basis vectors of the same orthonormal basis of the

[1]The laws of commutativity, associativity, and absorption are fulfilled.

underlying vector space. In that case, we can write $p_1 = \sum_{i_1} |k_{i_1}\rangle\langle k_{i_1}|$ and $p_2 = \sum_{i_2} |k_{i_2}\rangle\langle k_{i_2}|$ where the ket vectors $|k_i\rangle$ form an orthonormal basis. If two projectors commute then their join corresponds to the union of the respective one-dimensional oprators $|k_{i_j}\rangle\langle k_{i_j}|$ and their meet to the intersection. Thus, all projectors over a given orthonormal basis form a Boolean logic. This is affirmed by the following theorem.

Theorem 4.1 (Foulis-Holland) *Let L be an orthomodular lattice and a, b, c in L such that any one of them commutes with the other two. In this particular case the distributivity law holds.*

The following quotation from [Mar77] summarizes the main idea of quantum theory: '*Quantum theory is simply the replacement in standard probability theory of event-as-subset-of-a-set (abelian, distributive) by event-as-subspace-of-a-vector-space (non-abelian, non-distributive).*'

Chapter 5

Quantum Retrieval

In this chapter we develop basic concepts for mapping database tuples and relational calculus queries to the formalism of quantum mechanics and quantum logic (see Table 3.1). In this way, we extend the power of classical database query processing by dealing with probability values in order to support retrieval queries as well as proximity queries. By exploiting quantum logic on projectors we are able to construct complex queries. Please note, that in quantum logic projectors are combined to new projectors *before* any measurement w.r.t. an object takes place. Thus, a projector is capable to embody the complete semantics of every query.

Following, we distinguish between *categorical* and *ordinal* attributes.

5.1 Categorical Attributes

Categorical data are data on which no meaningful order exists. In our example, the different paint techniques (`oil, pencil, watercolor`) may be regarded as categorical data. The main idea of our quantum mapping is to bijectively assign each categorical value to exactly one basis vector:

Definition 5.1 (mapping categorical values) *A categorical value cv of a domain D with* $|D| = n$ *is expressed by a vector of a predefined basis of* $\mathbb{R}^n$*. The vector space* $\mathbb{R}^n$ *is spanned by the predefined set of n orthonormal basis vectors* $|c\rangle$ *where each* $|c\rangle$ *corresponds bijectively to a value* $c \in D$*.*

Next, we define projectors for select-queries.

Definition 5.2 (mapping categorical select-queries) *Let* $C \subseteq D$ *contain the required categories of a select-condition. Such a condition is expressed by the projector* $p_C = \sum_{c \in C} |c\rangle\langle c|$*.*

Example 5.1 The basis vectors $|oil\rangle = (1,0,0)^t$, $|pencil\rangle = (0,1,0)^t$, and $|watercolor\rangle = (0,0,1)^t$ represent three different paint techniques whereas the condition demanding oil or pencil paint technique is encoded by $|oil\rangle\langle oil| + |pencil\rangle\langle pencil|$.

Since all possible projectors p_C are based on the same basis they commute to each other. In consequence, following Theorem 4.1, negation, conjunction and disjunction on those projectors altogether constitute a Boolean logic. These logical operations correspond to set operations on the respective sets of orthonormal categorical basis vectors: $\neg p_C = p_{D \setminus C}, p_{C_1} \wedge p_{C_2} = p_{C_1 \cap C_2}$ and $p_{C_1} \vee p_{C_2} = p_{C_1 \cup C_2}$.

The following theorem shows that quantum measurement (Postulate 3) on categories yields same evaluation results as evaluating corresponding classical database select-queries.

Theorem 5.1 (measuring categorical values) *The measurement result of a projector p_C on a categorical vector $|cv\rangle$ is given by*

$$\langle cv|p_C|cv\rangle = \begin{cases} 1 & : cv \in C \\ 0 & : otherwise. \end{cases} \tag{5.1}$$

Proof

$$\langle cv|p_C|cv\rangle \;=\; \langle cv| \left(\sum_{c \in C} |c\rangle\langle c| \right) |cv\rangle = \sum_{c \in C} \langle cv|c\rangle\langle c|cv\rangle$$

Due to orthonormality of the basis vectors $|c\rangle$ we can write $\langle cv|c\rangle = \delta(cv, c)$ where δ is the Kronecker delta. That is, the measurement yields the value 1 only if $cv \in C$ holds. Otherwise, we obtain the value 0. $\square$

The theorem shows that our formalism supports complex select-queries as known from classical database theory on single-attribute categorical values obeying the rules of Boolean logic.

Remark: Our mapping allows for a state a probabilistic superposition of an ensemble E of categorical values by linearly combining the respective basis vectors ($|E\rangle = \sum_{e \in E} \sqrt{P_e}|e\rangle$) with their probabilities P_e. The measurement ($\langle E|p_C|E\rangle = \sum_{e \in E \cap C} P_e$) provides the summed up probability values of the matched categories.

5.2 Ordinal Attributes

Next, we introduce the mapping of ordinal attribute values and queries to our formalism. A domain of values is called *ordinal* if there is a meaningful order on its values. Thus, we are interested in distinguishing comparisons between two values which are close neighbors from those which lie far away from each other.

Initially, we assume a non-negative, ordinal numerical $a \in [0, \infty]$ as given to be mapped to a state vector. Please recall that state vectors need to be normalized. Therefore, we cannot directly map attribute values to a one-dimensional ket vector. Instead we need at least two dimensions. A two-dimensional quantum system in the field of quantum computation is called a *qubit* (*qu*antum *bit*). Since every normalized linear combination of two basis vectors $|0\rangle = (1, 0)^t$ and $|1\rangle = (0, 1)^t$ is a valid qubit state vector we can encode infinitely many ordinal attribute values. That is, we take advantage of the superposition principle of quantum mechanics. Please notice that no more than two vectors can be encoded as independent (orthogonal) state vectors within a one-qubit system.

Definition 5.3 (mapping ordinal values to qubit states) *The normalized qubit state $|a\rangle$ for a database value $a \in [0, \infty]$ is defined by*

$$a \mapsto |a\rangle = \frac{1}{\sqrt{a^2 + 1}} \begin{pmatrix} 1 \\ a \end{pmatrix}.$$

Thus, the database value is expressed by the normalized ratio between the two basis vectors $|0\rangle$ and $|1\rangle$.

Definition 5.4 (mapping ordinal select-queries) *In accordance with Def. 5.3, a select-condition with an associated non-negative numerical constant* c *is expressed by the projector $p_c = |c\rangle\langle c|$.*

Computing the degree of matching between a qubit state $|a\rangle$ and a select-condition $p_c = |c\rangle\langle c|$ by quantum measurement yields

$$\langle a|p_c|a\rangle = \frac{(1+ac)^2}{(a^2+1)(c^2+1)} \tag{5.2}$$

which equals the squared cosine of their enclosed angle. We obtain a proximity value near to zero (orthogonality in the geometric interpretation) only if one value is very high whereas the other one equals zero. Figure 5.1 depicts the corresponding graph for proximity values obtained from comparing two values from $[0\ldots7]$. Please notice that the dashed isolines for the proximity value 0.95 diverge from the diagonal with increasing attribute values. That is, the measurement is more sensitive to differences between small values than to those between large values.

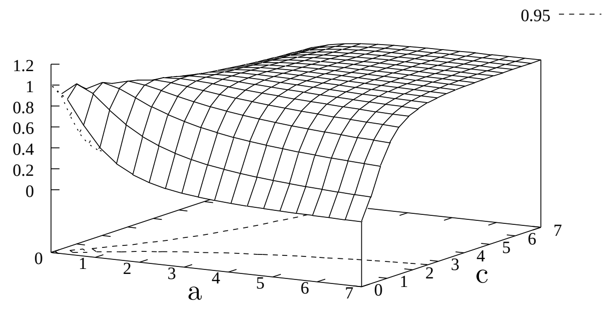


Figure 5.1: Graph for $\frac{(1+ac)^2}{(a^2+1)(c^2+1)}$

Next, we introduce a user-defined bijection $f : dom(A) \rightarrow [0, \infty]$ on the ordinal domain of an attribute A which we apply before quantum encoding following Def. 5.3 is performed. This gives the user a means to assign some meaningful semantics to resulting proximity values. As a positive side effect, such a mapping enables us to encode non-numerical values.

Example 5.2 (user-defined value mapping) In our introduced example, we want to encode the eight centuries from the 13th to the 20th. First, these ordinal values are bijectively mapped to the integers 0 to 7 ('13th' $\rightarrow$ 0, ..., '20th' $\rightarrow$ 7). Second, we map those integers to qubit state vectors producing measured proximity values dependent solely on the difference $d = |a - c|$. Thus, they represent the absolute error between attribute value a and a query condition constant c. Such a symmetry is achieved by linearly mapping the eight values to angles from 0 to $\pi/2$ realized by applying the function $f : a \mapsto \tan a\pi/16$. Figure 5.2 depicts the geometry of that mapping. Since the measurement value between two qubit states equals the squared cosine of the enclosed angle the tangent mapping produces a measurement result

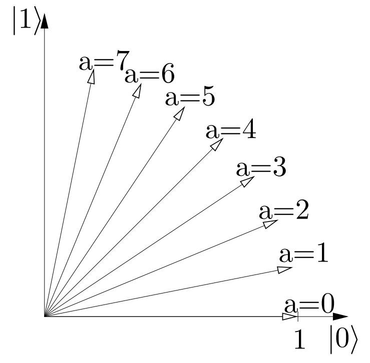


Figure 5.2: Equi-angular mapping

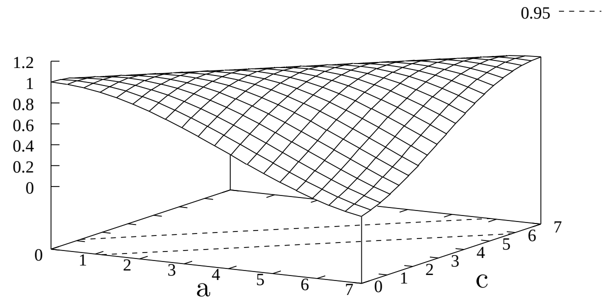


Figure 5.3: Graph for $\frac{(1+f(a)f(c))^2}{(f(a)^2+1)(f(c)^2+1)}$ and $f(x) = \tan x\pi/16$

given by $\cos^2 d\pi/16$. The corresponding measurement graph is depicted in Figure 5.3. Please notice that the dashed isolines are now parallel to the diagonal. That is, the measurement is equally sensitive to differences between small values and between large values.

Applying a user-defined bijection does not enable the simulation of any arbitrary similarity function $s(a,b)$ on ordinal values $a, b \in dom(A)$. Instead, several restrictions on $s(a,b)$ are implicitly stated. Due to the computation of the squared cosine of the enclosed angle we obtain three restrictions on $s(a,b)$:

1. $\forall a, b \in dom(A) : s(a,b) \in [0,1]$,
2. $\forall a \in dom(A) : s(a,a) = 1$, and
3. $\forall a, b \in dom(A) : s(a,b) = s(b,a)$.

Let the vectors for three ordinal values $a, b, c \in dom(A)$ lie on a plane[1] and the angles α, β, γ be the corresponding angles. Then, we can recompute the enclosed angle between $|a\rangle$ and $|b\rangle$ by

$$|\alpha - \beta| = \arccos \sqrt{s(a,b)}.$$

[1]Since we use a one-qubit system.

Taking three ordinal values of a qubit system into account, we obtain an additional restriction:

$$\begin{aligned} s(b,c) &= \cos^2(\beta - \gamma) \\ &= \cos^2 |\arccos \sqrt{s(a,c)} \pm \arccos \sqrt{s(a,b)}|. \end{aligned}$$

Thus, only similarity function fulfilling these restriction can be simulated by mapping it to a qubit system.

One disadvantage of the ordinal mapping is the missing support of disjunction and conjunction on the same attribute. For example, there is no way to express the condition '$(century = 15th) \vee (century = 16th)$' in a single-qubit system. The disjunction of conditions with different constants corresponds to the `join` operation which involves the computation of the vector space closure. As result, we obtain the projector $I = |0\rangle\langle 0| + |1\rangle\langle 1|$ which corresponds to the `true`-statement in Boolean logic. Analogously, the conjunctive combination of conditions with different constants produces a vector subspace containing just the origin as intersection of the respective vector subspaces (`meet`). The resulting `null`-matrix corresponds to the `false`-statement in Boolean logic.

So far, we showed that our mapping of categorical values supports the complete Boolean logic but due to the orthogonality of the mapped values there is no support of proximity queries. Contrarily, quantum processing of ordinal values supports the notion of proximity but fails to support conjunction and disjunction. Later on, we discuss this aspect in more detail.

5.3 Multi-Attribute Tuples

A typical database tuple contains more than one attribute value. Therefore, we have to adapt our mapping to the multi-attribute case. A multi-attribute tuple can be regarded as a composite quantum system. Adopting Postulate 4, we use the tensor product for constructing multi-attribute state vectors.

Definition 5.5 (database tuple as tensor product of single-attribute states) *Assume, a database tuple $t = (a_1, \ldots, a_n)$ contains n attribute values and $|a_1\rangle, \ldots, |a_n\rangle$ are their respective state vectors (regardless whether categorical or ordinal). Then, the ket vector*

$$|t\rangle = |a_1\rangle \otimes \ldots \otimes |a_n\rangle = |a_1..a_n\rangle$$

represents tuple `t`.

A single-attribute select-condition $A_j = c$ on a multi-attribute-tuple must be prepared accordingly. Thus, a single-attribute condition $|c\rangle\langle c|$ needs to be combined with all orthonormal basis vectors (expressed by the identity operator) of the non-restricted attributes.

Definition 5.6 (single-attribute select-condition) *Assume, $A_j = c$ is a select-condition on attribute A_j. Its projector p_c expressing the condition against an* `n`*-tuple is given by*

$$p_c = \underbrace{I \otimes \ldots \otimes I}_{(j-1)\times} \otimes |c\rangle\langle c| \otimes \underbrace{I \otimes \ldots \otimes I}_{(n-j)\times}.$$

The following measurement formula yields the measurement value for a given database tuple $|t\rangle{=}|a_1..a_n\rangle$.

$$\begin{aligned}
& \langle a_1..a_n|\underbrace{I\otimes\ldots\otimes I}_{(j-1)\times}\otimes|c\rangle\langle c|\otimes\underbrace{I\otimes\ldots\otimes I}_{(n-j)\times}|a_1..a_n\rangle \\
= \quad & \langle a_1|I|a_1\rangle\ldots\langle a_{j-1}|I|a_{j-1}\rangle\langle a_j|c\rangle * \\
& \langle c|a_j\rangle\langle a_{j+1}|I|a_{j+1}\rangle\ldots\langle a_n|I|a_n\rangle = \langle a_j|c\rangle\langle c|a_j\rangle. \qquad (5.3)
\end{aligned}$$

This formula equals the measurement of the single-attribute case. That is, the computation of the measurement becomes very easy since we can completely ignore non-restricted attributes.

5.4 Equality-Conditions

Equality-conditions require value equivalence of different attributes of the same type. Again, we distinguish between equality-conditions on categorical values and on ordinal values. The main idea is to construct a projector which refers to the vector subspace minimally containing all possible pairs of equal attribute values.

Definition 5.7 (mapping of a categorical equality-condition) *The equality between two categorical values $|c_1\rangle$ and $|c_2\rangle$ of a two-attribute tuple $|c_1c_2\rangle$ with $c_1, c_2 \in D$ is expressed by the projector*

$$p_{cc} = \sum_{c\in D}|cc\rangle\langle cc|.$$

The measurement of two categories yields the value 1 on equality and 0 otherwise.

Example 5.3 The equality-condition on two paint technique attributes is given by $|oil\ oil\rangle\langle oil\ oil|{+}|pencil\ pencil\rangle\langle pencil\ pencil|{+}|watercolor\ watercolor\rangle\langle watercolor\ watercolor|$.

Constructing a projector for the equality of two ordinal values is more complicated. The representation of a two-value-tuple (a_1, a_2) is given by the state

$$|a_1a_2\rangle = \frac{1}{\sqrt{a_1^2+1}\sqrt{a_2^2+1}}\begin{pmatrix}1\\a_2\\a_1\\a_1a_2\end{pmatrix} \Leftrightarrow \begin{matrix}|00\rangle\\|01\rangle\\|10\rangle\\|11\rangle.\end{matrix}$$

Here, we show also the bitcode representation of the corresponding canonical basis vectors. For a state with $a_1 = a_2$ we require the equivalence of the components $|01\rangle$ and $|10\rangle$. Therefore, both are combined into one normalized query vector: $(|01\rangle + |10\rangle)/\sqrt{2}$. Thus, the subspace of $(a_1{=}a_2)$-vectors is spanned by the orthonormal vectors $|00\rangle, (|01\rangle + |10\rangle)/\sqrt{2}$, and $|11\rangle$.

Definition 5.8 (mapping of an ordinal equality-condition) *The equality between two ordinal values $|a_1\rangle$ and $|a_2\rangle$ of a two-attribute tuple $|a_1a_2\rangle$ is expressed by the projector*

$$p_{aa} = |00\rangle\langle 00| + \frac{(|01\rangle + |10\rangle)(\langle 01| + \langle 10|)}{2} + |11\rangle\langle 11|.$$

Measuring an ordinal two-attribute tuple $|a_1a_2\rangle$ on equality using the two-attribute equality projector p_{aa} yields

$$\langle a_1a_2|p_{aa}|a_1a_2\rangle = \frac{a_1^2a_2^2 + \frac{(a_1+a_2)^2}{2} + 1}{a_1^2a_2^2 + a_1^2 + a_2^2 + 1}. \quad (5.4)$$

The result equals the value 1 if and only if a_1 equals a_2. Otherwise, we obtain a value smaller than 1 but greater than 0.5. The function converges to the value 0.5 if one value is zero whereas the other one grows towards infinity. Figure 5.4 depicts the graph when we apply the tangent encoding as described in Example 5.2.

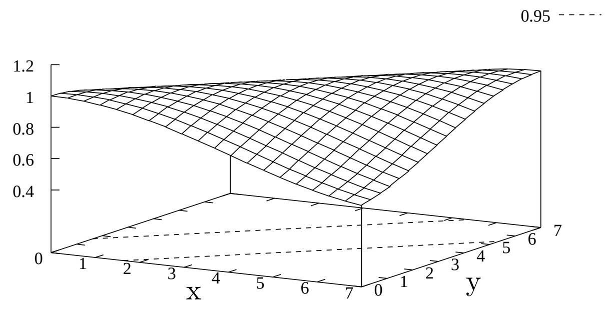


Figure 5.4: Graph for equi-angular ordinal values

Besides the two-attribute case, an equality-conditions can require the equivalence of more than two attributes. Whereas this generalization is obvious for categorical attributes this is not the case for ordinal ones. The procedure to construct the ordinal equality-subspace is analogous to the two-attribute case. All vectors of the canonical basis sharing the same number of ones in their bitcodes (e.g. the three-attribute ket vectors $|001\rangle$, $|010\rangle$, and $|100\rangle$ share exactly only one 1 in their bitcode) need to be combined into one basis vector of the equality subspace.

Definition 5.9 (mapping of a general ordinal equality-condition) *Assume, an n-attribute qubit state is given. The condition requiring the equality of all n ordinal single-attribute qubit states is expressed by the projector*

$$p = \sum_{i=0}^{n} |b_i\rangle\langle b_i|$$

with $|b_i\rangle = 1/\sqrt{\binom{i}{n}} \sum_j |b_i^j\rangle$ which groups all 2^n-dimensional vectors $|b_i^j\rangle$ of the canonical basis containing exactly i ones in their bitcodes into one normalized subspace basis vector.

Example 5.4 The projector for a three-attribute equality-condition is given by $p = |b_0\rangle\langle b_0| + |b_1\rangle\langle b_1| + |b_2\rangle\langle b_2| + |b_3\rangle\langle b_3|$ with

$$\begin{aligned}
|b_0\rangle &= |000\rangle \\
|b_1\rangle &= (|001\rangle + |010\rangle + |100\rangle)/\sqrt{3} \\
|b_2\rangle &= (|011\rangle + |101\rangle + |110\rangle)/\sqrt{3} \\
|b_3\rangle &= |111\rangle
\end{aligned}$$

The measurement value of an `n`-attribute equality-condition w.r.t. an arbitrary state $|a_1..a_n\rangle$ is given by

$$\sum_{i=0}^{n} \frac{\sum_j \langle a_1..a_n|b_i^j\rangle\langle b_i^j|a_1..a_n\rangle}{\binom{i}{n}}.$$

Analogously to constant-select-conditions, equality-conditions can be extended by non-restricted attributes using the tensor product.

5.5 Negation, Conjunction and Disjunction

A complex condition of the relational calculus is constructed by recursively applying conjunction, disjunction and negation on atomic conditions. Database disjunction, conjunction, and negation have their counterparts in quantum logic. That is, for combining two projectors conjunctively we apply the `meet` operator returning a new projector. Analogously, disjunction corresponds to the `join` operator and the negation of a condition is related to the negation of a projector. Despite dealing with probability values, quantum logic behaves like Boolean logic if involved projectors do commute.

Negation: The following theorem relates the negation of projectors to a measurement result.

Theorem 5.2 (negation) *Assume, a projector p_c expressing an arbitrary condition `c` is given. The measurement of its negation $p_{\neg c}$ on a database tuple $|t\rangle$ equals the subtraction of the non-negated measurement from 1:*

$$\langle t|p_{\neg c}|t\rangle = 1 - \langle t|p_c|t\rangle. \tag{5.5}$$

Proof Exploiting the definition of quantum negation and a state vector, we obtain

$$\langle t|p_{\neg c}|t\rangle \quad = \quad \langle t|I - p_c|t\rangle = \langle t|I|t\rangle - \langle t|p_c|t\rangle = 1 - \langle t|p_c|t\rangle.$$

□

Quantum negation extends Boolean negation. However, if a measurement returns a probability value between 0 and 1 then the effect may be surprising. For example, assume an attribute A of the three-valued ordinal domain $\{a, b, c\}$ is given. Surprisingly, as shown in Table 5.1, the negated condition $\neg(A = b)$ does *not* equal the condition $(A = a) \vee (A = c)$. Instead, that condition yields the *dissimilarity* between the attribute value and the value `b`. Thus, the measurement value of the ordinal value `a` is smaller than 1. This effect is the direct consequence of dealing with proximity of values.

Conjunction: Since we already discussed disjunction and conjunction on the *same* ordinal attribute we assume here conditions to be combined with disjoint sets of restricted attributes. Thus, they do commute and constitute, therefore, a Boolean algebra.

Theorem 5.3 (conjunction of disjoint conditions)
Let $p_a = p_a^1 \otimes \ldots \otimes p_a^n$ be a projector on n attributes and k restrictions on the attributes

query condition	database value		
	`a`	`b`	`c`
$A = b$	0.75	1	0.75
$A = a \vee A = c$	1	0.75	1
$\neg(A = b)$	0.25	0	0.25

Table 5.1: Negation and proximity values using the tangent encoding

$\{a_1, .., a_k\} \subseteq [1..n]$ *with*

$$p_a^i = \begin{cases} \textit{an } a_i\textit{-restriction} & : i \in \{a_1, .., a_k\} \\ I & : \textit{otherwise} \end{cases}$$

and $p_b = p_b^1 \otimes \ldots \otimes p_b^n$ *be a further projector with* l *restrictions on the attributes* $\{b_1, .., b_l\} \subseteq [1..n]$

$$p_b^i = \begin{cases} \textit{a } b_i\textit{-restriction} & : i \in \{b_1, .., b_l\} \\ I & : \textit{otherwise} \end{cases}$$

and $\{a_1, .., a_k\} \cap \{b_1, .., b_l\} = \emptyset$. *Their conjunction yields the projector* $p_{a \wedge b} = p_{a \wedge b}^1 \otimes \ldots \otimes p_{a \wedge b}^n$ *with*

$$p_{a \wedge b}^i = \begin{cases} \textit{an } a_i\textit{-restriction} & : i \in \{a_1, .., a_k\} \\ \textit{a } b_i\textit{-restriction} & : i \in \{b_1, .., b_l\} \\ I & : \textit{otherwise} \end{cases}$$

Proof The `meet` operation is defined over the intersection of the corresponding subspaces. Thus, we obtain following derivation

$$\begin{aligned} p_a \wedge p_b &= (p_a^1 \otimes \ldots \otimes p_a^n) \wedge (p_b^1 \otimes \ldots \otimes p_b^n) \\ &= (p_a^1 \wedge p_b^1) \otimes \ldots \otimes (p_a^n \wedge p_b^n) \\ &= p_{p_a^1(\mathbf{H}) \cap p_b^1(\mathbf{H})} \otimes \ldots \otimes p_{p_a^n(\mathbf{H}) \cap p_b^n(\mathbf{H})}. \end{aligned}$$

Due to the disjointness $\{a_1, .., a_k\} \cap \{b_1, .., b_l\} = \emptyset$ the vector space of every attribute restriction is intersected with $\mathbf{H}$ producing identical restrictions. Thus, all restriction are simply taken over. □

Remark: Theorem 5.3 deals with select-conditions only. Obviously, as long as the disjoint clause is fulfilled the theorem applies analogously if equality-conditions are involved.

Computing the measurement on a database tuple $|t\rangle$ yields

$$\langle t|p_{a \wedge b}|t\rangle = \langle t|p_a|t\rangle\langle t|p_b|t\rangle \tag{5.6}$$

due to the rule $\langle a_1 b_1 | a_2 b_2 \rangle = \langle a_1 | a_2 \rangle \langle b_1 | b_2 \rangle$. Thus, the measured results for conjunctively combined disjoint projectors are simply multiplied. This conforms the probabilistic conjunction of independent events.

Example 5.5 Our introduced example query *'retrieve all oil paintings showing evening twilight painted about in 16th century'* combines conjunctively a categorical (`technique`), a text retrieval (`desc`), and a proximity (`century`) query. Since these query components are independent from each other their respective measurement results w.r.t. a certain XML-document are simply multiplied.

Special cases: In Theorem 5.3, we assumed conditions with restrictions being disjoint on attribute level. Thus, we obtain commuting projectors and therefore a Boolean logic. Next, we introduce four special cases where the demand for disjointness is abandoned due to commutativity of the conjunctively combined projectors.

1. *categorical attributes:* Since categorical values are bijectively mapped to *orthonormal* basis vectors overlapping conditions do always commute.
2. *select-condition and select-condition:* Two projectors which express select-conditions and overlap on some ordinal attributes do commute only if the overlapping select-conditions require the same select-constant. This includes also the negated case. That is, following the notation from Theorem 5.3, we require $\forall i \in \{a_1,..,a_k\} \cap \{b_1,..,b_l\} : p_a^i \in \{|c\rangle\langle c|, \neg|c\rangle\langle c|\} \wedge$
$p_b^i \in \{|c\rangle\langle c|, \neg|c\rangle\langle c|\}$ with a fixed ordinal value c.
3. *equality-condition and equality-condition:* Two conjunctively combined equality-conditions overlapping on some ordinal attributes can be merged to one large equality-condition. For example, we can prove that $p_{a_1=a_2} \wedge p_{a_2=a_3}$ equals $p_{a_1=a_2=a_3}$. This rule is used to remove overlapping ordinal equality-conditions.
4. *equality-condition and select-condition:* An equality-condition conjunctively combined with an overlapping ordinal select-condition (or its negation) is transformed to non-overlapping select-conditions. We can prove that the rule $(p_{a_1=a_2} \wedge p_{a_1=c}) \implies (p_{a_1=c} \wedge p_{a_2=c})$ always holds.

Thus, requiring disjoint conditions with the exceptions of the four listed special cases guarantees that the corresponding projectors commute and, therefore, constitute a Boolean logic.

Disjunction: From Chapter 4 we know that quantum logic respects the de Morgan law. Therefore, we can compute the measurement for the disjunction of projectors over conjunction and negation and obtain

$$\begin{aligned}\langle t|p_{a\vee b}|t\rangle &= 1-(1-\langle t|p_a|t\rangle)(1-\langle t|p_b|t\rangle) \\ &= \langle t|p_a|t\rangle + \langle t|p_b|t\rangle - \langle t|p_{a\wedge b}|t\rangle.\end{aligned} \tag{5.7}$$

The discussed semantics of disjoint conjunction, disjunction, and negation obey the rules of probability theory for independent events. Furthermore, the logical operations on disjoint projectors equal the algebraic product and the algebraic sum being a t-norm and a t-conorm of fuzzy-logic [Zad88], respectively. However, our theory is richer with respect to the semantics of underlying conditions. For example, Formulas 5.6 and 5.7 are valid on non-overlapping conditions only. The problem of violated idempotence of the algebraic product does not occur in our theory (see the special cases): The `meet` and the `join` operation collapse the combination of equal conditions (second special case) automatically to one condition fulfilling the demand of idempotence.

5.6 Commuting Quantum Query Processing

In this section we define a new query language and a feasible algorithm to process corresponding queries. The language is recursively built from atoms and formulas.

Definition 5.10 (commuting quantum query language) *The commuting quantum query language is based on a given relation schema of n attributes $A_1, .., A_n$. Assume, function* type: $\{A_1, \ldots, A_n\} \mapsto \{cat, ord\}$ *returns for every attribute its type (ordinal or categorical). An* atom *is defined to be one of three alternatives:*

1. *A select-condition '$A_i = c$' with constant c is an atom.*
2. *An equality-condition '$A_{i_1} = \ldots = A_{i_k}$' on k attributes of the same type is an atom.*
3. *A set-containment on a categorical attribute '$A_i \in C$' is an atom.*

A set of atoms At is called commuting *if the following condition holds:*

$$\begin{aligned}&\forall atom_1 \in At : \forall atom_2 \in At :\\&\forall A_i \in involved(atom_1) : \forall A_j \in involved(atom_2) :\\&(atom_1 \neq atom_2 \wedge A_i = A_j) \implies (type(A_i) = cat) .\end{aligned}$$

The function involved *returns the set of all attributes restricted by an atomic condition*[2].

A commuting quantum query on a commuting atom set At is recursively defined as follows:

1. *Every atom of At is a query.*
2. *If q is a quantum query then $\neg q$ is a query.*
3. *If q_1 and q_2 are two queries then $(q_1 \wedge q_2)$ and $(q_1 \vee q_2)$ are queries.*

As result, we obtain commuting query expressions for which the rules of Boolean algebra apply.

Evaluation algorithm: The general goal is to evaluate a given commuting quantum query with respect to a tuple $(v_1, .., v_n)$. We will show that such an evaluation does not require complex algorithms from linear algebra. Instead, our algorithm is based on simple boolean transformations and basic arithmetic operations.

A direct evaluation of conjunction and disjunction by applying Formula 5.6 and 5.7 is not possible since the formulas were defined on expressions with *non-overlapping* conditions. However, our language allows overlaps as long as the underlying atom set is commuting. Our

[2]Please note, that the notion of a commuting set respects the four special cases. Instead of explicitly allowing the third and the fourth special case, we assume such conditions to be transformed accordingly in advance in order to obtain a commuting set.

idea is to apply Boolean rules to transform expressions with overlapping conditions into non-overlapping ones. Actually, we need to resolve overlaps just on ordinal attributes. Categorical literals (negated or non-negated atomic conditions) produce boolean values which are correctly respected by Formula 5.6 and 5.7 regardless whether they overlap or not. The algorithm for transforming an arbitrary commuting quantum query e is given in Figure 5.5.

```
input:  commuting quantum language expression $e$
output:  $e$ without ordinal overlaps

(1)  transform expression $e$ into
     disjunctive normal form $\hat{x}_1 \vee \ldots \vee \hat{x}_m$
     where $\hat{x}_i$ are conjunctions of literals
(2)  simplify expression $e$ by applying
     idempotence and invertibility[3]rules
(3)  if there is an overlap on an ordinal
     attribute between some conjunctions $\hat{x}_i$ then
  (3a) let $o$ be a literal of an attribute
       common to at least two conjunctions
  (3b) replace all conjunctions $\hat{x}_i$ of $e$
       with $(o \wedge \hat{x}_i) \vee (\neg o \wedge \hat{x}_i)$
  (3c) simplify $e$ by applying idempotence,
       invertibility, and absorption and obtain
       $e = (o \wedge \hat{x}_1) \vee \ldots \vee (o \wedge \hat{x}_{m_1}) \vee$
       $(\neg o \wedge \hat{x}_{m_1+1}) \vee \ldots \vee (\neg o \wedge \hat{x}_{m_2})$
  (3d) replace $e$ with $(o \wedge e_1) \vee (\neg o \wedge e_2)$ where
       $e_1 = \hat{x}_1 \vee \ldots \vee \hat{x}_{m_1}, e_2 = \hat{x}_{m_1+1} \vee \ldots \vee \hat{x}_{m_2}$
  (3e) continue with step (3) for $e_1$ and $e_2$
(4)  transform innermost disjunctions to
     conjunctions and negations by applying
     de-Morgan-law
```

Figure 5.5: Transformation algorithm to resolve overlaps

Analyzing the transformation result, we observe that the subformulars of the innermost disjunctions (the leaves of the corresponding tree) are mutually non-overlapping on ordinal attributes[4] before we apply the fourth step. That is, we can apply Formula 5.7. All other disjunctions are based on *exclusive* subformulas (generated by step (3d)). That is, we can simply drop the conjunction term from Formula 5.7 before we apply this formula. That is, we simply add the scores. Since, furthermore, all conjunctions are based on non-overlapping subformulas Formula 5.6 directly applies. The fourth step is to simplify arithmetic calculation of multiple disjunctions.

[3]invertibility: $a \vee \neg a = 1, a \wedge \neg a = 0, \neg\neg a = a$

[4]Otherwise the algorithm would not have stopped.

Since overlaps on ordinal attributes are now resolved we can directly apply Formulas 5.1, 5.3, 5.4, 5.5, 5.6, and 5.7 in order to evaluate the query against a database tuple.

Next, we demonstrate the evaluation using an example. The atoms of our example query are presented in Table 5.2. The condition on the textual description of a painting is a text retrieval query, the condition on the century of its creation is a proximity query, and the conditions on the three different painting techniques are classical database (categorical) queries. Their evaluations w.r.t. to a tuple $t = (t_d t_c t_t)$ are shown on the right side. For the text retrieval evaluation we simply take the squared cosine of the angle between the corresponding text vectors. The proximity evaluation is performed in accordance with Example 5.2.

In our example query given in Figure 5.6, we search for paintings which show a crucifixion or for watercolor paintings. The crucifixion should be painted with oil if created in the 17th century, otherwise with pencil. Figure 5.6 demonstrates the transformation algorithm step by step and the final arithmetic evaluation formula with respect to a given tuple $t = (t_d t_c t_t)$.

condition	evaluation
d: `desc='crucifixion'`	$d^t = \langle t_d \vert 'crucifixion' \rangle^2$
c: `century='17th'`	$c^t = \cos^2 \vert 4 - t_c \vert \cdot \pi/16$
t_1: `technique='oil'`	$t_1^t = \begin{cases} 1 & : t_t = 'oil' \\ 0 & : otherwise \end{cases}$
t_2: `technique='pencil'`	$t_2^t = \begin{cases} 1 & : t_t = 'pencil' \\ 0 & : otherwise \end{cases}$
t_3: `technique='watercolor'`	$t_3^t = \begin{cases} 1 & : t_t = 'watercolor' \\ 0 & : otherwise \end{cases}$

Table 5.2: Atomic conditions and their evaluations w.r.t. tuple $t = (t_d t_c t_t)$

$$\begin{tikzcd}
(d \wedge ((c \wedge t_1) \vee (\neg c \wedge t_2))) \vee t_3 \arrow[d, "(1)(2)"'] \\
(c \wedge d \wedge t_1) \vee (\neg c \wedge d \wedge t_2) \vee t_3 \arrow[d, "(3a)(3b)(3c)"', "o = c"] \\
(c \wedge d \wedge t_1) \vee (c \wedge t_3) \vee (\neg c \wedge d \wedge t_2) \vee (\neg c \wedge t_3) \arrow[d, "(3d)"'] \\
(c \wedge ((d \wedge t_1) \vee t_3)) \vee (\neg c \wedge ((d \wedge t_2) \vee t_3)) \arrow[d, "(4)"'] \\
(c \wedge \neg (\neg (d \wedge t_1) \wedge \neg t_3)) \vee (\neg c \wedge \neg (\neg (d \wedge t_2) \wedge \neg t_3))
\end{tikzcd}$$

arithmetic evaluation w.r.t. tuple t:

$$c^t \left(1 - \left(1 - d^t t_1^t\right)\left(1 - t_3^t\right)\right) +$$
$$\left(1 - c^t\right)\left(1 - \left(1 - d^t t_2^t\right)\left(1 - t_3^t\right)\right)$$

Figure 5.6: Example transformations and arithmetic evaluation

5.7 Quantum Queries

Our commuting quantum query language introduced in Definition 5.10 is based on a commuting set of atoms. Different select-conditions on a common ordinal attribute are not allowed[5]. Thus, we cannot express queries like `desc='crucifixion'` $\vee$ `desc='martyr'` or `century` $\leq$ `'14th'` (`century` $=$ `'13th'` $\vee$ `century` $=$ `'14th'`). As shown in Section 5.2, a disjunction of non-commuting conditions would produce the `true`-statement and its conjunction the `false`-statement. This results from the way how quantum matching is performed: In general, quantum processing supports a matching of a value (normalized vector) against a set of 'orthonormal' values (projector). In case of a categorical attribute all values are orthonormal per definition. In the ordinal case this does not hold. Thus, the query value set is restricted to contain just one value.

What should be our target semantics of a disjunction like `desc='crucifixion'` $\vee$ `desc='martyr'`? We require:

1. The disjunction restricted to crisp truth (boolean) values should obey the laws of boolean logic (commutativity, associativity, boundary condition, monotonicity, and idempotence).
2. Every returned score should be interpretable as a probability value produced by a commuting query.

Both requirements are fulfilled by applying the standard fuzzy t-conorm `max`. Analogously, the conjunction corresponds to the standard t-norm `min`.

Given a query being non-commuting due to different conditions on common ordinal attributes, our idea is to single out such conditions and to evaluate them in isolation from quantum evaluation by applying the `min`/`max`-functions. Since `min`/`max` return always one input score as output score the output can be interpreted as a value produced by just one common ordinal condition. Therefore, we can regard all non-commuting conditions on a common attribute as just *one* ordinal condition multiply used. In this way, we obtain a commuting query and can therefore apply the transformation algorithm proposed in Figure 5.5. Next, we define syntax and evaluation of a quantum query.

Definition 5.11 (quantum query language) *The quantum query language is based on a given relation schema of n attributes* $A_1, .., A_n$. *Assume, function* type: $\{A_1, \ldots, A_n\} \mapsto \{cat, ord\}$ *returns for every attribute its type (ordinal or categorical). An* atom *is defined to be one of four alternatives:*

1. *a select-condition '*$A_i = c$*' with constant c;*
2. *an equality-condition '*$A_{i_1} = \ldots = A_{i_k}$*' on k attributes of same type;*
3. *a set-containment '*$A_i \in C$*';*
4. *a range-condition on an ordinal attribute '*$A_i \leq c$*' or '*$A_i \geq c$*';*

A quantum query on an atom set At is recursively defined as follows:

[5]Please note, due to categorical conditions our language is at least as powerful as a classical database query language.

1. *Every atom of At is a quantum query.*
2. *If q is a quantum query then $\neg q$ is a quantum query.*
3. *If q_1 and q_2 are two quantum queries then $(q_1 \wedge q_2)$ and $(q_1 \vee q_2)$ are quantum queries.*

Again, we assume special cases three and four to be transformed accordingly in advance. That is, any condition conjunctively combined with an overlapping equality condition is copied to the second attribute and overlapping equality conditions are merged.

Definition 5.12 (conflict attribute A, conflict set, and conflict class) *An ordinal attribute A of a quantum query is called* conflict attribute *if:*

$$\begin{aligned}
&\exists atom_1, atom_2 \in At(atom_1 \neq atom_2 \wedge \\
&A = involved(atom_1) \wedge A = involved(atom_2)) \vee \\
&\exists C('A \in C' \in At) \vee \\
&\exists c('A \leq c' \in At) \vee \exists c('A \geq c' \in At).
\end{aligned}$$

The set of all conflict attributes of a quantum query is called conflict set. *The following equivalence relation which relates two arbitrary attributes A and B of a conflict set if*

1. *A and B are identical: $A = B$ or*
2. *A and B are connected (possibly transitively) by a equality condition: $'A = B' \in At$*

produces a set of equivalence classes called conflict classes.

For evaluation of a quantum query we have to check if its conflict set is empty or not. An empty conflict set means a commuting query wich is processed as described in the previous section .

Otherwise, we replace every atom which involves a conflict attribute of a conflict class with a conflict substitute. A conflict substitute represents exactly one conflict class and is from now on regarded as one normal atom. Applying this substitution, we obtain a commuting quantum query which is processed as described in previous section .

For example, see the query given in Figure 5.7. The ordinal attributes `desc` and `century` are conflict attributes. Each of them represents a conflict class (`cc1=`{`desc`} and `cc2=`{`century`}). Replacing conflicting atoms with their conflict substitutes provides the commuting query illustrated in Figure 5.8. Given the evaluation $cc1^t$, $cc2^t$, and $technique^t$ with respect to a tuple $t = (t_d t_c t_t)$, we obtain from our evaluation algorithm the formula $cc1^t * cc2^t * technique^t$.

The remaining open question is how to evaluate conflict substitutes.

Definition 5.13 (evaluation of conflict substitutes) *For a given conflict class of a non-commuting quantum query perform the following three steps:*

1. *Find the largest subqueries which do not contain any conflict attribute of the conflict class. If such a subquery is* conjunctively *connected to the remaining formula then replace it with the* `true`*-statement and with a* `false`*-statement in case of a disjunction.*

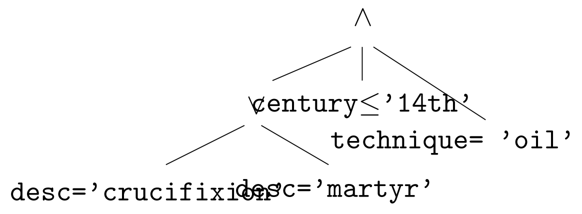


Figure 5.7: Non-commuting example query

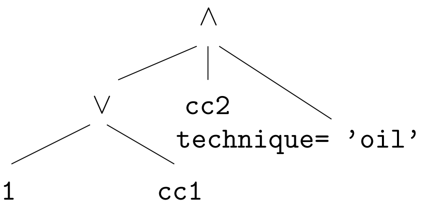


Figure 5.8: Commuting example query

2. *Simplify the formula by applying border conditions and idempotence.*
3. *Replace every conjunction with the* `min`*-function and every disjunction with the* `max`*-function.*

If we apply these steps to the example substitutes we obtain

$$\begin{aligned} cc1^t = max(\quad & eval^t \quad (desc =' crucifixion'), \\ & eval^t \quad (desc =' martyr')) \end{aligned}$$

$$cc2^t = eval(century \leq' 14th')$$

Last but not least, we have to define the evaluation of a set-containment and a range-condition.

Definition 5.14 (evaluation of an ordinal set-containment) *The evaluation of a set-containment $'A \in C'$ with respect to a tuple t is defined by*

$$eval^t('A \in C') = \max_{c \in C} eval^t('A = c').$$

Definition 5.15 (evaluation of an ordinal range-condition) *The evaluation of a range-condition*
$'A \leq c'$ with respect to a tuple t is defined by

$$\begin{aligned} eval^t('A \leq c') &= \max_{v \in dom(A) \wedge v \leq c} eval^t('A = v') \\ &= \begin{cases} 1 & : A^t \leq c \\ eval^t('A = c') & : otherwise \end{cases}. \end{aligned}$$

The evaluation of $'A \geq c'$ is analogously defined.

5.8 Incorporating Weights into Conjunction and Disjunction

During constructing a complex query from ordinal conditions there is often a need to put more weight on one condition than on the other one. Consider for example a query where we search for a person with a long nose and short hair, and we are more sure about the nose than the hair. Thus, the nose condition should stronger influence the result. Similar to Fagin's weighting schema [Fag98], we state for a weight $\theta \in [0,1]$ on an operand of a conjunction or a disjunction following requirements:

1. *equi-weighted case:* Equal weights on both operands produce the same result as the unweighted case.

2. *zero weight:* A zero-weighted operand should not have any influence on the result. That is, the result is the same as if that operand is completely removed.

3. *linearity:* The weighting formula interconnects linearly both extremes (equi- and zero weight).

4. *Boolean logic:* Incorporating weights into disjunction and conjunctions does not invalidate the rules of Boolean logic[6].

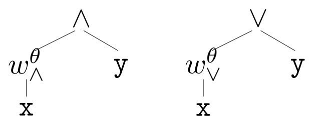


Figure 5.9: Weighted conjunction and disjunction

In our approach, we assume at least one operand having full weight ($\theta = 1$). Weighting the remaining operand with $\theta \leq 1$ is realized by applying a special weighting operator on that operand. As illustrated in Figure 5.9, we introduce the weighting operator $w_\wedge^\theta$ above operand `x` directly before performing conjunction, and $w_\vee^\theta$ before disjunction, respectively. Following, we define a weighting fulfilling the first three requirements.

Definition 5.16 (weighted conjunction and disjunction) *Assume, condition* `x` *of a conjunction '$x \wedge y$' and a disjunction '$x \vee y$' is given a weight $\theta \in [0,1]$. With respect to a given tuple t we define the following evaluations by taking into account our conjunction and disjunction evaluations (see Formulas 5.6 and 5.7):*

$$\begin{aligned} eval^t(w_\wedge^\theta(x) \wedge y) &= (1 - \theta(1 - x^t))y^t \\ eval^t(w_\vee^\theta(x) \vee y) &= \theta x^t + y^t - \theta x^t y^t \end{aligned}$$

where x^t stands for $eval^t(x)$ and y^t for $eval^t(y)$.

[6]Please note, that Fagin's weighting schema does not fulfill that requirement.

Please note, that following rules hold:

$$\begin{aligned}
w_{\wedge}^{\theta_1}\left(w_{\wedge}^{\theta_2}(x)\right) &= w_{\wedge}^{\theta_1 * \theta_2}(x)\\
w_{\vee}^{\theta_1}\left(w_{\vee}^{\theta_2}(x)\right) &= w_{\vee}^{\theta_1 * \theta_2}(x)\\
w_{\wedge}^{\theta}(x) &= \neg w_{\vee}^{\theta}(\neg x)\\
w_{\vee}^{\theta}(x) &= \neg w_{\wedge}^{\theta}(\neg x)
\end{aligned}$$

Theorem 5.4 (Boolean logic rules for evaluations including weights) *Weighting an operand of a disjunction or a conjunction as defined in Definition 5.16 obeys the laws of Boolean logic if identical conditions are equally weighted.*

Proof[sketch] Following Theorem 4.1 we need to show that we are able to realize evaluations according to Definition 5.16 by commuting projectors. Our main idea is to double the number of underlying dimensions by introducing a shadow vector space. That is, every tuple $|t\rangle$ is mapped to

$$|t\rangle \mapsto \left| \begin{array}{c} t \\ 0 \end{array} \right\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes |t\rangle.$$

Due to the zero multiplier the shadow space has no impact on any evaluation. Analogously, we double the space of any projector of an atomic condition

$$p \mapsto \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes p.$$

The basic idea of our weighting is to move query 'energy' between the original space and the shadow space. If the weight of an operator decreases then the projector approaches the null matrix for disjunction and the identity matrix for conjunction, respectively:

$$\begin{aligned}
w_{\wedge}^{\theta}(p_x) &= rot_{1-\theta} * (I - p_x) * rot_{1-\theta}^{T} + p_x\\
w_{\vee}^{\theta}(p_x) &= rot_{\theta} * p_x * rot_{\theta}^{T}
\end{aligned}$$

where

$$rot_{1-\theta} = \left(\begin{array}{cc|cc}
1-\theta & 0 & -\sqrt{2\theta-\theta^2} & 0\\
 & \ddots & & \ddots\\
0 & 1-\theta & 0 & -\sqrt{2\theta-\theta^2}\\
\hline
\sqrt{2\theta-\theta^2} & 0 & 1-\theta & 0\\
 & \ddots & & \ddots\\
0 & \sqrt{2\theta-\theta^2} & 0 & 1-\theta
\end{array}\right)$$

$$rot_{\theta} = \left(\begin{array}{cc|cc}
\theta & 0 & -\sqrt{1-\theta^2} & 0\\
 & \ddots & & \ddots\\
0 & \theta & 0 & -\sqrt{1-\theta^2}\\
\hline
\sqrt{1-\theta^2} & 0 & \theta & 0\\
 & \ddots & & \ddots\\
0 & \sqrt{1-\theta^2} & 0 & \theta
\end{array}\right)$$

Please notice, that rot_{θ} and $rot_{1-\theta}$ as well as their transposes are orthonormal matrices which just rotate 'energy' between the original and the shadow dimensions. Furthermore, their sizes can be adapted to the number of required dimensions. It can be easily shown that for a database tuple $|t\rangle$ and two commuting projectors $w^\theta_\wedge(p_x)$ (or $w^\theta_\vee(p_x)$) and p_y

$$\begin{aligned}\left\langle \begin{array}{c} t \\ 0 \end{array} \middle| w^\theta_\wedge(p_x) \wedge p_y \middle| \begin{array}{c} t \\ 0 \end{array} \right\rangle &= (1-\theta(1-x^t))y^t \text{ and} \\ \left\langle \begin{array}{c} t \\ 0 \end{array} \middle| w^\theta_\vee(p_x) \vee p_y \middle| \begin{array}{c} t \\ 0 \end{array} \right\rangle &= \theta x^t + y^t - \theta x^t y^t\end{aligned}$$

hold where x^t stands for $eval^t(x) = \langle t|p_x|t\rangle$ and y^t for $eval^t(y) = \langle t|p_y|t\rangle$. But are $w^\theta_\wedge(p_x)$ (or $w^\theta_\vee(p_x)$) and p_y really commuting projectors? We know that

1. if p is a projector and rot an orthonormal matrix then $rot * p * rot^T$ is also a projector,
2. if p is a projector then $I - p$ is also a projector, and
3. adding two projectors which represent disjoint vector spaces produces again a projector.

As result, it turns out that $w^\theta_\wedge(p_x)$ and $w^\theta_\vee(p_x)$ are projectors. Next, we have to show that given two commutative projectors a weighting produces again commutative projectors. The projector's definition is based on a set of orthonormal basis vectors. Checking commutativity can be broken down to the underlying basis vectors of a projector. Commutativity of two projectors is satisfied if and only if their basis vector sets are subsets from one common set of orthonormal basis vectors. That is, basis vectors from two commutative projectors are either identical or orthogonal. Applying our weighting operator on a doubled basis vector $|x\rangle$ equals

$$rot_\theta \left| \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes x \right\rangle = \left| \begin{pmatrix} \theta \\ \sqrt{1-\theta^2} \end{pmatrix} \otimes x \right\rangle .$$

Furthermore, regarding the scalar product we obtain

$$\left\langle \begin{pmatrix} \theta_1 \\ \sqrt{1-\theta_1^2} \end{pmatrix} \otimes x_1 \middle| \begin{pmatrix} \theta_2 \\ \sqrt{1-\theta_2^2} \end{pmatrix} \otimes x_2 \right\rangle = \\ \left\langle \begin{array}{c} \theta_1 \\ \sqrt{1-\theta_1^2} \end{array} \middle| \begin{array}{c} \theta_2 \\ \sqrt{1-\theta_2^2} \end{array} \right\rangle \langle x_1|x_2\rangle.$$

Thus, we conclude that orthonormality ($\langle x_1|x_2\rangle$=0) is unaffected by any weighting.

But what about original identical basis vectors ($\langle x_1|x_2\rangle$=1). For keeping identity we simply require the same weight. Let us consider a basis vector restricting an ordinal attribute. Due to our dealing with conflicting ordinal attributes (see previous chapter) different conditions on a common ordinal attribute do not occur for a given database tuple[7]. Thus, we can guarantee commutativity if the same ordinal condition is weighted equally. We require the same for identical conditions on categorical attributes. Since different conditions on a same categorical attribute are mutually orthogonal we can ignore that case.

But how to check weight equality for a given query? In order to check equal weight on same conditions it is enough to check the path within the query tree from that condition to the

[7]Different conditions with weight are dealt with the min / max operators. Interestingly, it can be easily shown that applying our weighting operator w^θ produces the same weighting as Fagin's formula with the substitution $\theta_1 = \frac{1+\theta}{2}$ and $\theta_2 = \frac{1-\theta}{2}$.

current logical operator. Please remember, that conjunction/disjunction on commutative projectors corresponds to intersection/union on their sets of basis vectors. Thus, the weighting of a condition remains unaffected by any conjunction/disjunction. On the path we select just the weighting operators and the negations. Applying our weighting rules, we can transform all conjunctive weight operators into disjunctive one, multiply subsequent weights and simplify double negations. At the end we obtain as a *weight normal form* a path of alternating negations and disjunctive weighting operators which can be easily compared with the one of other branches. □

5.9 The Quantum Query Language QQL

In this section we briefly discuss the transition of our language towards the query language QQL. The transition is similar similar to the transition of propositional logic to first order logic.

- *Relations:* So far, we evaluated queries against a given database tuple. As an extension, we introduce variables. Analogously to the relational domain calculus we assume finite relations with tuples being available in form of relation predicates. Relation predicates are Boolean although they can contain ordinal attribute values.

- *Quantors:* Originally, the quantors $\exists$ and $\forall$ are defined on Boolean values. However, due to ordinal conditions and weighting of our language we obtain truth values from $[0, 1]$. The question is how to evaluate such quantors in this case? From predicate logic we know how to transform any logical formula into the prenex normal form. As result, the quantors do not have any impact on the evaluation of single tuples. In order to evaluate an $\exists$ quantor on a variable we apply the maximum function to the evaluation results of different variable-to-value-substitutions. Analogously, we use the minimum for the $\forall$ quantor.

Next, we define our complete query language QQL.

Definition 5.17 (QQL) *The quantum query language QQL is based on a set $\{R_i\}$ of relation schemata. Each contains a subset of $\{A_1, \ldots, A_n\}$. Assume, function* type: $\{A_1, \ldots, A_n\} \mapsto \{cat, ord\}$ *returns for every attribute its type (ordinal or categorical) and a set of variables $\{X_i\}$ is given. An* atom *is defined to be one of five alternatives:*

1. *a relation predicate '$R_i(X_{j_1}, \ldots, X_{j_k})$';*
2. *a select-condition '$X_i = c$' with constant c;*
3. *an equality-condition '$X_{i_1} = \ldots = X_{i_k}$' on k variables of same type;*
4. *a set-containment '$X_i \in C$';*
5. *a range-condition on an ordinally bound variable '$X_i \leq c$' or '$X_i \geq c$';*

A quantum query on an atom set At is recursively defined as follows:

1. *Every atom of At is a quantum query.*

2. *If q is a quantum query then $\neg q$ is a quantum query.*

3. *If q_1 and q_2 are two quantum queries then $(q_1 \wedge q_2)$ and $(q_1 \vee q_2)$ are quantum queries.*

4. *If q_1 and q_2 are two quantum queries and $\theta \in [0,1]$ a weight constant then $(w^\theta_\wedge(q_1) \wedge q_2)$, $(q_1 \wedge w^\theta_\wedge(q_2))$, $(w^\theta_\vee(q_1) \vee q_2)$, and $(q_1 \vee w^\theta_\vee(q_2))$ are quantum queries.*

5. *If q is a quantum query and X is a free variable of q then $(\exists X)(q)$ and $(\forall X)(q)$ are quantum queries.*

The query is called valid if no multiply used atom is differently weighted (see Section 5.8).

For a finite query processing we require any valid query to be a *safe* query. As result of query processing we obtain variable-to-value substitutions together with their corresponding score values. The score value must be higher than zero.

Please notice, that if we restrict our language to cardinal conditions we obtain the relational domain calculus[8]. However, our query language extends the relational domain calculus by dealing with uncertainty and proximity. Thus, our language incorporates concept from information retrieval into a classical database language basing on one unifying theoretic framework.

[8]Remark: a smaller-than and a greater-than condition on a classical database attribute can be simulated by a set-containment condition on a cardinality attribute.

Chapter 6

Conclusion and Outlook

In this work, we mapped traditional database queries basing on Boolean logic to the formalism of quantum mechanics and logic. As result, we obtain a new view of the process of database query processing. There is a rich set of techniques from linear algebra available in order to try to solve database problems. Furthermore, we used that formalism in order to extend the expressiveness power of database queries to cope seamlessly with proximity and retrieval search terms. A retrieval search can be simply incorporated into our formalism by adding a retrieval vector space via the tensor product to a given quantum vector space.

Quantum measurement results can be regarded as probability values. Furthermore, one interesting result is that quantum conjunction, disjunction and negation conforms the rules of probability theory. In contrast to competing approaches, e.g. fuzzy databases, a quantum query represented as projector does not just combine given non-discrete truth values but embodies entirely the underlying query semantics.

Our work describes mainly theoretical results. In order to realize these results it is not necessary to completely simulate quantum systems. Instead, a relatively small algorithm (not presented here) can be developed on the basis of the formulas 5.2, 5.4, 5.5, 5.6, and 5.7.

In future, we plan to include the quantifiers $\exists$ and $\forall$ into our quantum formalism in order to cover the complete power of relational calculus. Analogously to our similarity relational calculus approach using fuzzy logic described in [SS04], we will develop a complete query system which enables us to combine retrieval, proximity and database queries in a user-friendly and efficient way.

Bibliography

[BG73] R. Bellman and M. Giertz. On the Analytic Formalism of the Theory of Fuzzy Sets. *Information Science*, 5:149–156, 1973.

[Bol94] N. Bolloju. A Calculus for Fuzzy Queries on Fuzzy Entity-Relationship Model. Technical Report 94/26, Department of Information Systems at the City Polytechnic of Hong Kong, 1994.

[BP95] P. Bosc and O. Pivert. SQLf: A Relational Database Language for Fuzzy Querying. *IEEE Transactions on Fuzzy Systems*, 3(1):1–17, February 1995.

[BR99] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* ACM Press, Essex, England, 1999.

[BvF81] E. Beltrametti and B.C. van Fraassen, editors. *Current Issues in Quantum Logic.* Plenum Press, 1981.

[CMPT00] Paolo Ciaccia, Danilo Montesi, Wilma Penzo, and Alberto Trombetta. Imprecision and user preferences in multimedia queries: A generic algebraic approach. In K.-D. Schewe and B. Thalheim, editors, *FoIKS: Foundations of Information and Knowledge Systems, First International Symposium, FoIKS 2000, Burg, Germany, February 14-17, 2000*, volume 1762 of *Lecture Notes in Computer Science*, pages 50–71. Springer, 2000.

[CNC00] I. Chuang, M. A. Nielsen, and I. L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.

[Cod71] E. F. Codd. A Database Sublanguage Founded on the Relational Calculus. In *ACM SIGFIDET Workshop on Data Description, Access and Control*, pages 35–61, nov 1971.

[DD97] C. J. Date and H. Darwen. *A Guide to the SQL Standard.* Addison-Wesley, Reading, MA, 4 edition, 1997.

[Dir58] P. Dirac. *The Principles of Quantum Mechanics.* Oxford University Press, 4th edition, 1958.

[Fag98] R. Fagin. Fuzzy Queries in Multimedia Database Systems. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 1-3, 1998, Seattle, Washington*, pages 1–10. ACM Press, 1998.

[FR97] N. Fuhr and T. Rölleke. A Probabilistic Relational Algebra for the Integration of Information Retrieval and Databases Systems. *ACM Transactions on Information Systems (TOIS)*, 15(1):32–66, January 1997.

[Gle57] A. Gleason. Measures on the Closed Subspaces of a Hilbert Space. *Journal of Mathematics and Mechanics*, 6:885 – 893, 1957.

[GMPC98] J. Galindo, J. M. Medina, O. Pons, and J. C. Cubero. A Server for Fuzzy SQL Queries. In T. Andreasen, H. Christiansen, and H. L. Larsen, editors, *Flexible Query Answering Systems, Third International Conference, FQAS'98, Roskilde, Denmark, May 13-15*, volume 1495 of *Lecture Notes in Computer Science*, pages 164–174. Springer, 1998.

[Gru99] J. Gruska. *Quantum Computing*. McGraw-Hill, 1999.

[GUP05] J. Galindo, A. Urrutia, and M. Piattini. *Fuzzy Databases: Modeling, Design and Implementation*. Idea Group Publishing, 2005.

[Loc85a] P.F. Lock. Connections among quantum logics, part 1: Quantum propositional logics. *International Journal of Theoretical Physics*, 24:43–53, 1985.

[Loc85b] P.F. Lock. Connections among quantum logics, part 2: Quantum event logics. *International Journal of Theoretical Physics*, 24:55–61, 1985.

[Mai83] D. Maier. *The Theory of Relational Databases*. Computer Science Press, Rockville, MD, 1983.

[Mar77] A. R. Marlow. *Mathematical Foundations of Qantum Theory*, chapter Orthomodular Structures and Physical Theory. Academic Press, 1977.

[RP00] E. Rieffel and W. Polak. An introduction to quantum computing for non-physicists. *ACM Computing Surveys*, 32(3):330–335, 2000.

[SS02] N. Schulz and I. Schmitt. A Survey of Weighted Scoring Rules in Multimedia Database Systems. Preprint 7, Fakultät für Informatik, Universität Magdeburg, 2002.

[SS04] I. Schmitt and N. Schulz. Similarity Relational Calculus and its Reduction to a Similarity Algebra. In Dietmar Seipel and J. M. Turull-Torres, editors, *Third Intern. Symposium on Foundations of Information and Knowledge Systems (FoIKS'04), Austria, February 17-20*, volume 2942 of *Lecture Notes in Computer Science*, pages 252–272. Springer-Verlag Berlin Heidelberg, 2004.

[SSH05] Ingo Schmitt, Nadine Schulz, and Thomas Herstel. WS-QBE: A QBE-like Query Language for Complex Multimedia Queries. In Yi-Ping Phoebe Chen, editor, *Proceedings of the 11th International Multimedia Modelling Conference (MMM'05), Melbourne, Australia, January 12-14, 2005*, pages 222–229, Los Alamitos, CA, jan 2005. IEEE Computer Society Press.

[Tak93] Y. Takahashi. Fuzzy Database Query Languages and Their Relational Completeness Theorem. *IEEE Transaction on Knowledge and Data Engineering*, 5(1):122–125, February 1993.

[vN32] J. von Neumann. *Grundlagen der Quantenmechanik.* Springer Verlag, Berlin, Heidelberg, New York, 1932.

[vR79] C. J. van Rijsbergen. *Information Retrieval.* Butterworths, London, 1979.

[vR04] C. J. van Rijsbergen. *The Geometry of Information Retrieval.* Cambridge University Press, Cambridge, England, 2004.

[Zad88] Lofti A. Zadeh. Fuzzy Logic. *IEEE Computer*, 21(4):83–93, April 1988.

[Zie05] Martin Ziegler. Quantum Logic: Order Structures in Quantum Mechanics. Technical report, University Paderborn, Germany, 2005.