A Quantum Interpretation of the View-Update Problem

Christian Flender

Faculty of Science and Technology,
Queensland University of Technology,
Level 8 / 126 Margaret Street,
Brisbane QLD 4000, Australia,
Email: c.flender@qut.edu.au

Abstract

The ANSI-SPARC architecture was proposed as a hierarchical model for the implementation of Database Management Systems (DBMS). A separation of external user views and shared base relations (conceptual schema) constitutes their logical independence, i.e., external views are immune to changes of the conceptual schema. Moreover, users can customize their views independent of the conceptual schema. However, all updates to a base relation should be immediately reflected in all views that reference the base relation. Vice versa, if a view is updated, then the underlying base relation should reflect the change. Keeping views and base relations in sync came to be known as the view-update problem. This paper argues that view updates require the user to cause a change. Prior to a view update user and view are entangled. Entangled states cannot be reduced to factual states of user and base relation. It will be shown that the view-update problem arises due to a view update (causation) being irreducible to a functional mapping between base relation and view (causality).

Keywords: ANSI-SPARC architecture, view-update problem, quantum entanglement, causation, causality

1 Introduction

As early as 1971, the Data Base Task Group (DBTG) appointed by the Conference of Data Systems and Languages (CODASYL) proposed a general architecture for database systems (CODASYL 1971). This proposal already recognized the need for several levels of abstraction in order to deal with the diverging needs of database stakeholders such as end-users, application developers and administrators (DBAs). Already hierarchical or first-generation DBMS divided their system into schema and subschema. The former was meant to provide a view for DBAs who were mainly concerned with the definition of database names, types of records and components of these records. The schema was separated from the so called subschemas, the part of the database as seen by users or application programs. Then in 1975, with the rise of relational or second-generation DBMS, which can be traced back to E.F.Codd's influential paper in 1970 (Codd 1970), the American National Standards Institute's (ANSI) Standards Planning and Requirement Committee (SPARC) produced a similar multi-level architecture with the same purpose, i.e., the provision of an implementation independent layer to isolate programs from underlying representational issues (ANSI-SPARC 1975). The three-level ANSI-SPARC architecture is a *de facto* standard for the implementation of relational DBMS and its relevance to information systems continues to the present day.

The three-level architecture consists of an external level, a conceptual level, and an internal level (cf. Figure 1). The way users perceive data, their personalized view (top-down), belongs to the external level. The internal level abstracts from the way the operating system accesses data (bottom-up), i.e., via data structures and file organisations. The conceptual level provides a mapping between external and internal level and thereby mediates between customized user views and physical data implementation. Generally, at the conceptual level the concep-

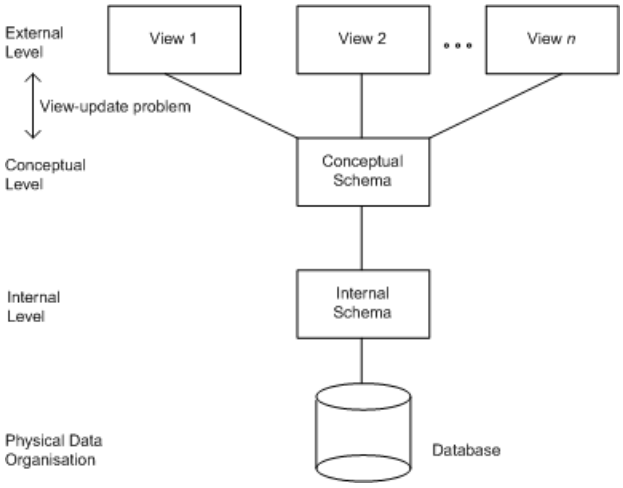

Figure 1: The ANSI-SPARC architecture divides a DBMS into three levels. At the top the external level consists of customized views and therefore interfaces to end-users and application developers. At the bottom the internal level abstracts from the physical data implementation. The conceptual level provides a mapping between external level and internal level and thereby mediates between views and physical data organisation.

tual schema constitutes a community view, i.e., an underlying representation shared by all external views. It describes base relations (entities), attributes, relationships and integrity constraints. Views at the external level derive from the conceptual schema. More precisely, views are the dynamic result of one or more relational operations, i.e., selection, projection,

Cartesian product, union and set difference, acting upon base relations. A separation of external user views and shared base relations constitutes their logical independence, i.e., external views are immune to changes of the conceptual schema. Users can customize their views at the external level independent of the conceptual schema. However, all updates to a base relation should be immediately reflected in all views that reference the base relation. Vice versa, if a view is updated, then the underlying base relation should reflect the change. Keeping views and base relations in sync came to be known as the view-update problem.

There are several approaches attempting to solve this problem of external-conceptual mapping (see related work in Section 4). However, none of these approaches has ever considered a view update at the external level as a quantum-like cause (conditions for change are necessary but not sufficient) being irreducible to a determinate functional mapping (conditions for change are necessary and sufficient) at the conceptual level. This article aims to offer a quantum interpretation of the view-update problem. It is argued that, prior to a view update, user and view are entangled (user and view cannot be separated). The change of a view itself is indeterminate, i.e., the state of a view is determined by the outcome of an update but not prior to it. Accordingly, changes being propagated to the conceptual level do not necessarily translate into determinate functional view mappings, i.e., the application of one or more relational operators (selection, projection, Cartesian product, union and set difference). Entangled states of user and view at the external level are irreducible to states of user and base relation at the conceptual level. Hence, quantum entanglement offers a reasonable explanation of the view-update problem.

The contribution of this paper is to show that the interface between user and DBMS is not as clear cut as the logical and physical independence of the ANSI-SPARC architecture might suggest. Although the ANSI-SPARC architecture claims to separate logic from physics, it cannot separate the user from the DBMS. User-view entanglement inseparably couples user and DBMS. A view derives from a base relation; a view update, however, often cannot be reduced to changes at the conceptual level. What came to be known as the view-update problem has a natural and reasonable explanation. Views are not independent from their implementation but rely on the user's causation for their realization.

The next section introduces the view-update problem and shows where change propagation from external to conceptual level breaks down. Then, in Section 3, user-view relationships will be shown to be entangled and therefore quantum-like. Section 4 discusses existing work related to the view-update problem, in particular related to indeterminate updates (causation) at the external level and determinate updates (causality) at the conceptual level. Finally, Section 5 concludes the article and gives an outlook towards future work.

2 The View-Update Problem

The view-update problem arises at the interface between views at the external level and base relations at the conceptual level (cf. Figure 1). The latter constitutes a shared representation accessible by multiple users and independent from storage space allocation and index structures. At the conceptual level, the DBMS manages base relations. A base relation is a named relation, e.g., *Staff*, corresponding to an entity in the real world. Concrete instances of a base relation, e.g., *Staff* member 001, are tuples and stored physically in the database. A view, on the other hand, is defined at the external level. It is the dynamic result of one or more relational operations[1], i.e., selection, projection, Cartesian product, union and set difference, acting upon a base relation to produce another relation, e.g., *Manager* as a subset of *Staff*. Hence, a view is a virtual or derived relation in the sense that it belongs to the external level and can be produced upon request by a particular user.

Views offer flexibility and security by hiding certain parts of the database from certain users, e.g., the view *Manager* doesn't contain information about board members. Furthermore, users can customize their information needs. The same data can be seen in different ways at the same time. Moreover, views can simplify complex operations upon base relations. Since changes of a base relation, e.g., adding attributes or tuples, do not necessarily require changes of views, they are considered to be logically independent. Nevertheless, all updates of a base relation should be immediately reflected in all views that reference the base relation. Vice versa, if a view is updated, then the underlying base relation should propagate the change. Here, the view-update problem arises due to the fact that there are view updates which do not translate into base relation updates so that, eventually, the changed view equals the view of a changed base relation. An update mapping does not always exist, and even when it does exist, it may not be unique (Codd 1975). Therefore, a change in the view may not be reflected unambiguously by equivalent changes in the base relation. In formal terms,

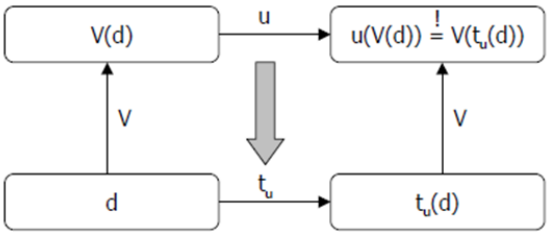

Figure 2: The view-update problem refers to the synchronization of external level and conceptual level. A view is derived from a base relation by means of operators as defined in relational algebra. If a user causes a view update this has to be propagated to the conceptual level such that the changed view equals the view projection of the changed base relation.

consider the state d of a base relation (cf. Figure 2). This state is projected onto the external level by a mapping V. V is functional, i.e., conditions for the mapping are causally necessary and sufficient. Effectively, V refers to one or more relational operations, i.e., selection, projection, Cartesian product, union and set difference, acting upon the base relation d to produce a view $V(d)$. If a user carries out a view update u, then this results in the changed view $u(V(d))$. Accordingly, at the conceptual level, there must be a base relation update t_u such that it reflects the change at the external level. Hence, the challenge is to find a transformation

$$u \rightarrow t_u, \tag{1}$$

[1]For a proof showing that operators of relational algebra are essentially equivalent in expressive power to relational calculus and thus first-order logic see (Codd 1970).

where $u(V(d)) = V(t_u(d))$. However, according to the view-update problem, $u \rightarrow t_u$ is indeterminate for arbitrary mappings V. Put in another way, there are view-updates u and mappings V for which external level and conceptual level cannot be synchronized, and, given u and V, there is no general rule for determining synchronizability.

The following example illustrates the view-update problem. Consider a simple base relation *Staff*. The current state of the base relation is d = *Staff*{(001,B1),(002,B2)}. The state d of the base relation is composed of two tuples; the first value of each tuple stands for the identification number of a staff member. The second entry represents the department number. A selection V = 'get *Staff* where depNo=B1' defines the mapping that generates a view. The view is meant to be a subset of all staff members, in particular managers working in department B1. Consequently, $V(d)$ = *Manager*{(001,B1)}.

Now assume a user updates this view. For instance, if he carries out u = 'update *Manager* set depNo=B3 where ID=001', then this will change the view such that $u(V(d))$ = *Manager*{(001,B3)}.

Accordingly, changes need to be done at the conceptual level. A transformation $u \rightarrow t_u$ is needed which satisfies the constraint $u(V(d)) = V(t_u(d))$ = *Manager*{(001,B3)}. However, there is no transformation $u \rightarrow t_u$ where $u(V(d)) = V(t_u(d))$ if the view update u 'interferes' with the view mapping V. In other words, the projection V defines a view $V(d)$ by operating upon a base relation d. If an update u upon $V(d)$ alters the view mapping V, then view and base relation cannot be synchronized. For example, consider the transformation for which the view update u is applied at the conceptual level such that t_u = u = 'update *Staff* set depNo=B3 where ID=001'. It follows that the base relation changes to $t_u(d)$ = *Staff*{(001,B3),(002,B2)}. However, $V(t_u(d))$ = *Manager*{$\emptyset$} and therefore $u(V(d)) \neq V(t_u(d))$. If view update u and view mapping V interfere, then there is no transformation $u \rightarrow t_u$ such that $u(V(d)) = V(t_u(d))$. Interference is nothing beyond reasoning. In fact, it is well known in quantum theory and thus the view-update problem suggests a quantum interpretation.

3 User-View Entanglement

Recently quantum formalisms have been adopted to problem descriptions outside of physics (Bruza, Lawless, van Rijsbergen, Sofge, Coecke & Clark 2008, Bruza et al. 2009). Examples related to databases are models in information retrieval (IR) (Piwowarski & Lalmas 2009*b*,*a*) or the design of part-whole relationships using the Entity-Relationship (ER) notation (Flender et al. 2009). One of the selling points of quantum models is their ability to represent effects like context-dependence and emergence (Kitto 2008). Phenomena like human-computer interactions are observer-relative and therefore context-dependent. An observation, action, or change, can be modelled as context. For instance, the meaning of a word like *Bat* depends on the context in which it is used. It might be understood as an animal or a sports utility dependent on its evocation (Bruza, Kitto, Nelson & McEvoy 2008). Similarily, emergent phenomena are often conceptualized as entangled or nonseparable states. For instance, an instance of a combined concept like *Pet Fish* can be modelled as an entangled or nonseparable state that emerges with dependence upon context (Aerts & Gabora 2005*b*,*a*). Experiments have shown that users generally associate instances like 'guppy' neither as a good example of the word *Pet* nor of the word *Fish*. However, their rating in the context of *Pet Fish* reveals a strong association. Therefore, 'guppy' is modelled as an emergent, entangled or nonseparable state of *Pet Fish*. Modelling emergent and context-dependent states requires operators which do not exist in relational algebra. The two operators relevant to the view-update problem are the measurement function and the Tensor product. The latter shall be introduced later in this section.

The quantum formalism defines an actualization function which corresponds to a measurement in quantum physics. This function is indeterminate in the sense that the resulting state of an observed system is determined by the outcome of the measurement but not prior to it. An observer causes the system to change (causation) but he will never be able to give necessary and sufficient conditions for a change to occur (causality). The probability involved is non-classical since there is no underlying deterministic assumption. Consequently, prior to a change or update, there are no necessary and sufficent conditions, i.e., the state in which a system will be found by chance was not determined at the time the change was made. In contrast, classical probabilistic functions assume a system to be in one or another state, i.e., the state of a system, though not yet determined, is in a definite state. Accordingly, relational operations as physically instantiated by a DBMS always presuppose views and base relations to be in a definite state. To make this more precise, we need to have a look at the quantum formalism, in particular vector spaces and operators.

A vector space is a multi-dimensional space. A Hilbert space is an abstract or infinite-dimensional vector space over the set of real or complex numbers. It is equipped with inner products, i.e., rules to measure distances and angles between vectors. Vectors have a magnitude denoting their relative size or length in space. Consider the orthonormal basis B of a two-dimensional Hilbert space (cf. Figure 3). For an orthonormal basis B, basis vectors are both normalized and orthogonal to each other. Two vectors[2] $|0\rangle$ and $|1\rangle$ are orthogonal if their angle is 90 ° or their cosine equals 0. The dot product or inner product $\langle 0|1\rangle$ returns a scalar. If vectors are normalized this scalar equates the cosine and hence measures the angle or

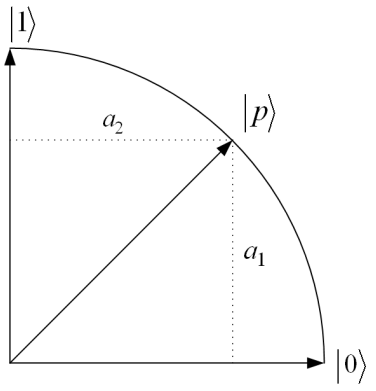

Figure 3: The figure shows a two-dimensional Hilbert space. It is equipped with inner products, i.e., rules to measure distances and angles between vectors. The orthonormal basis represents a view B which is composed of two values or qubits $|1\rangle$ and $|0\rangle$. After an update u, the view is in a state $|0\rangle$ where $u(V(d)) \neq V(t_u(d)$ or in a state $|1\rangle$ where $u(V(d)) = V(t_u(d))$.

[2]Note that besides (Brak-)ket notation ($\langle|$ and $|\rangle$) vectors can be written in terms of linear algebra or coordinate vectors.

distance between them. Basis vectors are normalized if their length equals 1 and so they are called unit vectors. A vector of arbitrary length can be divided by its length to create a unit vector. The length or magnitude of a vector is, according to Pythagoras, the square root of the sum of all squared vector components.

Each vector $|p\rangle$ can be written as a linear combination of orthogonal vectors.

$$|p\rangle = a_1|0\rangle + a_2|1\rangle, \tag{2}$$

where $|a_1|^2 + |a_2|^2 = 1$. The vector $|p\rangle$ represents a superposition or potentiality state. To illustrate the relationship between superposition states and actual states (eigenstates), consider the classical example of wave-particle complementarity.

Generally, matter and energy exhibit both wave-like and particle-like properties but not both at the same time, i.e., not within the same context. In different contexts or experimental arrangements some matter seems more particle-like than wave-like. With reduced values of energy (change of context) the same matter will be more likely to show wave-like qualities than particle-like properties. All the information about a particle is encoded in its wave function, which is analogous to the amplitude of a wave at each point in space. This function evolves according to a differential equation (the Schrödinger equation) and so gives rise to interference. Interference occurs when the interaction of two or more waves, e.g., one wave representing observer and the other one standing for the observed system, influences their direction of propagation characterized by crests and troughs. When two or more waves reach the same point in space at the same time, they either add up (the crests arrive together which is called in-phase) or cancel each other out (the crest from one wave meets a trough from another wave which is called out-of-phase). The state of a wave-like property is called superposition or potentiality state and represented as a vector $|p\rangle$. Its linear combination, the superposition or addition of two or more states, resembles an interference pattern typical of waves. If an observer measures the location

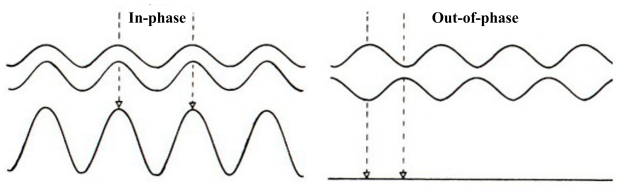

Figure 4: Interference occurs when the interaction of two or more waves, e.g., one wave representing observer and the other one standing for the observed system, influences their direction of propagation characterized by crests and troughs. When two or more waves reach the same point in space at the same time, they either add up (the crests arrive together which is called in-phase) or cancel each other out (the crest from one wave meets a trough from another wave which is called out-of-phase).

of the particle encoded in $|p\rangle$, the wave-function will randomly collapse to a well-defined position, a state like $|1\rangle$ or $|0\rangle$ traditionally associated with particles. The act of measurement, or context, of a particular situation is modelled as a choice of basis where the eigenstate of the system is chosen from the set of possibilities in such a way that it is compatible with the choice of action, i.e., context, to be performed upon it. Expectation values $|a_1|^2$ and $|a_2|^2$ are related to the probability $P(|0\rangle)$ and $P(|1\rangle)$ of the system being eventually found in their respective eigenstates $|0\rangle$ and $|1\rangle$. Put in another way, the likelihood of any particular location, or eigenstate, equals the squared amplitude $|a_1|^2$ and $|a_2|^2$ of the wave-function in this location.

$$P(|0\rangle) = |a_1|^2 \text{ and } P(|1\rangle) = |a_2|^2 \tag{3}$$

How does this actualization function translate into the ANSI-SPARC architecture? As discussed in the previous section, external views constitute the interface to the user. Whether end-user or application developer, at one stage he or she will update a view. Although database operations are physically instantiated causal functions, the actual change requires causation. Causation means the user makes an effort to change a view for which he may give reasons, e.g., a customer wants a modified view on his data, the program code must be changed, etc. However, such reasons will never be causally necessary and sufficient. Causation is not causality. Therefore, prior to an update u, a view $V(d)$ is always superposed in a state $|p\rangle$. After an update u the view is in a state $|0\rangle$ where $u(V(d)) \neq V(t_u(d))$ or in a state $|1\rangle$ where $u(V(d)) = V(t_u(d))$. Now, according to the view-update problem, $u \rightarrow t_u$ is indeterminate for arbitrary mappings V. Put in another way, there are view-updates u and mappings V for which it cannot be determined whether a) $|p\rangle$ collapses to $|1\rangle$, i.e., there is a transformation $u \rightarrow t_u$ such that $u(V(d)) = V(t_u(d))$, or b) $|p\rangle$ collapses to $|0\rangle$, i.e., there are only transformations $u \rightarrow t_u$ such that $u(V(d)) \neq V(t_u(d))$. Therefore, we need a superposition state which cannot be reduced to $|0\rangle$ or $|1\rangle$. To explain this irreducible indeterminism inherent in the view-update problem, the user has to be modelled. A situation in which a user cannot give causally necessary and sufficient conditions for a view $V(d)$ or state $|p\rangle$ to be reduced to a state $|1\rangle$ where $u(V(d)) = V(t_u(d))$ or a state $|0\rangle$ where $u(V(d)) \neq V(t_u(d))$ refers to the entanglement of user and view.

Entanglement requires the combined system user *plus* view to be in a superposition state that is neither reducible to the user nor to the view. Prior to a view update user and view are entangled. Moreover, views are virtual relations derived from the conceptual schema, i.e., the community view. Therefore, views, though not reducible to base relations, depend on the conceptual schema nevertheless. Entangled states of user and view refer to one integrated whole. More technically, an entangled state of user plus view cannot be written as a Tensor product of the user's current state and the view's current state. The following paragraph formalizes this situation.

The Tensor product is an outer product of matrices for which states can be easily shown to be entangled. In contrast to the Cartesian product of relational algebra, the Tensor product operates within combined Hilbert spaces. Tensor products are outer products of vectors and generate states within combined systems. The dimensionality of a composite state space is I^J where I is the number of single-body systems, e.g., 2 for user and view, and J is the sum of vector components, e.g., 2 for success and failure. For instance, the tensor product of two vectors associated with two bases B_1 (user) and B_2 (view) operates within a four-dimensional vector space $B = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. Each vector ψ in this space is regarded as corresponding to a possible state of the associated pair of two-state systems. If such a state can be factorized into states of the single bases involved it is separable, i.e., it is a product state. However, it is easy to find vectors which

cannot be expressed as a tensor product of a pair of state vectors from the associated quantum systems. Such vectors are entangled states. In terms of the

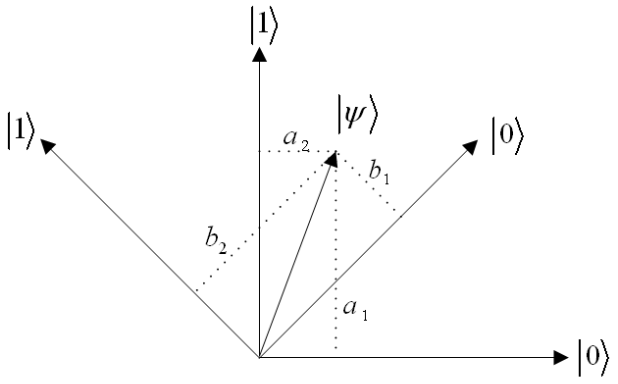

Figure 5: The figure shows a combined system composed of two qubit bases $B_1 = \{|0\rangle, |1\rangle\}$ (user) and $B_2 = \{|0\rangle, |1\rangle\}$ (view). Vectors ψ generated within that space represent possible states of the system. ψ is a superposition which is either a separable or an entangled state.

view-update problem, a product or separable state $|\psi\rangle$ refers to the combined state of user $|p_1\rangle$ and view $|p_2\rangle$ (or $V(d)$) for which the outcome of u has determined whether a) there is a transformation $u \rightarrow t_u$ such that $u(V(d) = V(t_u(d))$, or b) there are only transformations $u \rightarrow t_u$ such that $u(V(d) \neq V(t_u(d))$. If user B_1 and view B_2 disentangle, then ψ is separated. *A posteriori* the state of $u(V(d))$ is either $|0\rangle$ or $|1\rangle$ determined by the outcome of u.

$$\begin{aligned} |\psi\rangle &= |p_1\rangle \otimes |p_2\rangle \\ &= (a_1|0\rangle + a_2|1\rangle) \otimes (b_1|0\rangle + b_2|1\rangle) \\ &= a_1b_1|00\rangle + a_2b_1|10\rangle + a_1b_2|01\rangle + a_2b_2|11\rangle, \end{aligned} \tag{4}$$

where $|a_1b_1|^2 + |a_2b_1|^2 + |a_1b_2|^2 + |a_2b_2|^2 = 1$.

Prior to an update u, however, user and view are entangled. If $|\psi\rangle$ is entangled, there is no $|p_1\rangle$ and $|p_2\rangle$ such that $|\psi\rangle = |p_1\rangle \otimes |p_2\rangle$. In contrast to product states, entangled quantum states are not separable. If two or more quantum states entangle, they interact so that a nonseparable state emerges. The joint probability that emerges cannot be factorized into single probabilities. In physics, the empirical setting for this phenomenon involves two spatially separated (non-local) measuring devices performing spatially separated measurements. Eventually, one of the measurements disentangles B_1 (user) and B_2 (view) and the outcome determines the eigenstates in which the system will be found.

$$|\psi\rangle = x|00\rangle + y|11\rangle, \tag{5}$$

where $|x|^2 + |y|^2 = 1$.

In terms of the view-update problem, user B_1 and view B_2 interact in a way that it cannot be determined whether an update u applied to the view $V(d)$ translates into an update t_u applied to the base relation d such that $u(V(d)) = V(t_u(d))$. The entangled state $|\psi\rangle$ of the user-view whole cannot be reduced to a failed state $|0\rangle \in B_2$ (there is no transformation) or a successful state $|1\rangle \in B_2$ (there is a transformation).

Reconsider Equation 4 and Equation 5. Since $x > 0$ it follows that $a_1 > 0$ and $b_1 > 0$. Moreover, there is no coefficient for $|10\rangle$ and $|01\rangle$ and therefore $a_2 = 0$ or $b_1 = 0$. However, $y|11\rangle$ implies that $a_2 > 0$ and therefore it contradicts the fact that $a_2 = 0$. Consequently, there are no two states each belonging to one of the bases B_1 (user) and B_2 (view) and thus ψ is a nonseparable or entangled state.

4 Related Work

So far it has been shown that a quantum interpretation of the view-update problem gives a reasonable answer to the following questions. Why is there no rule for determining synchronizability? In other words, why is there no rule for determining whether view updates translate into base relation updates for arbitrary view mappings or not? A quantum interpretation of the view-update problem explains this indeterminism. User and view are entangled and therefore, prior to a view update, it is not determined whether this update translates into changes of a base relation or not. Due to the irreducibility of user and view there may be ambiguous or interfering changes of the base relation which cannot be determined prior to the user's update. Put in another way, the user's causation is irreducible to relational operators and thus to causal functions as physically instantiated by the DBMS. Moreover, there is no independence between user and DBMS. The physical realization of a view update depends on the user's causation. The remainder of this section shall discuss causation and causality in relation to existing work on the view-update problem.

First attempts to solve the view-update problem can be traced back to the early 1980s (Bancilhon & Spyratos 1981, Dayal & Bernstein 1982). Several classes have been defined for which views are theoretically updatable, theoretically not updatable, and partially updatable. However, given an update u, for arbitrary V, there is no general rule that determines synchronizability (see (Furtado & Casanova 1985) for a survey on updating views). An update mapping $u \rightarrow t_u$ does not always exist, and even when it does exist, it may not be unique (Codd 1975). A change in the view may not be reflected unambiguously by equivalent changes in the base relation. Nevertheless, many attempts to synchronize external and conceptual level have been made; attempts to compute $u \rightarrow t_u$ such that $u(V(d)) = V(t_u(d))$. To deductively derive unambiguous transformations $u \rightarrow t_u$, view updates u are usually constrained to certain views $V(d)$, in particular those views $V(d)$ that can be produced by the view projection V. Views are derived or virtual relations and therefore restricted to their creation using relational operators of relational algebra. Stringent conditions have to be introduced in order for a view update to translate into a base relation update such that the changed view equals the view of a changed base relation. Otherwise, undesirable side effects may occur effectively rendering external level and conceptual level inconsistent. Constraints can be realized by limiting updates to views satisfying several conditions. For instance, Dayal and Bernstein (1982) propose conditions under which a view update translates into a base relation update (Dayal & Bernstein 1982). Conditions were derived in terms of base relation instances and view instances using functional dependencies, keys, and subset constraints. The authors define simple syntactic translation procedures and derive checkable conditions that characterize when the translations produced by these procedures will satisfy the various correctness criteria. Accordingly, most commercial relational DBMS implementing the Standard Query Language (SQL) constrain updates to particular views. For instance,

views should be defined using simple queries involving a single base relation and the primary key or a candidate key of that base relation. Hence, updates are not allowed through views involving the Cartesian product over multiple base relations. Moreover, views derived from complex operations like aggregate functions and groupings are prohibited.

Other approaches have attempted to compute a transformation $u \rightarrow t_u$ using view complements (Bancilhon & Spyratos 1981). View complements represent missing data sources being complementary to the data of a view. Such complements are used in conjunction with a view update u in a way sufficient to deductively derive changes t_u of the base relation d. Hence, complementary information is exploited to translate view updates to base relation updates. The base relation can be reconstructed from the view and its complement. However, complements are not unique and the choice of an optimal complement requires deductive rules which haven't been found. In fact, it has been argued that there is no way to compute optimal complements within relational algebra if views derive from projection operators (Lechtenbörger & Vossen 2003).

In summary, deductive attempts to solve the view-update problem enforce constraints upon possible view changes. In this way, a DBMS defines requirements, metaphorically speaking a straitjacket, for updates u. In terms of the ANSI-SPARC architecture, requirements are imposed bottom-up. Physically instantiated functions dictate the range of possible views. Hence, restricting the user in his possibilities is not really a solution to, or explanation of, the view-update problem. Rather it is an educational program that enforces the user to speak the language of relational algebra, and thus first-order logic. The fact that, for practical purposes, such languages get a face-lift, e.g., SQL and other declarative or procedural languages, doesn't avoid the enforcement of constraints upon possible view changes.

Another interesting attempt to synchronize views and base relations uses abductive reasoning (Kakas & Mancarella 1990). Abductive reasoning is a kind of backward reasoning, i.e., the inverse of modus ponens. According to modus ponens: $A \rightarrow B$, $A \vdash B$. For instance, if someone is a human (A), then he is mortal (B). Christian is a human, therefore he is mortal. In contrast to this forward chaining which starts with the antecedent condition A, abductive reasoning starts with an observation B. It looks for possible explanations of this observation. According to a given theory, there are several explanations of someone being mortal, e.g., he is a man or she is a woman. In terms of the view-update problem, a changed view $u(V(d))$ represents the conclusion or observation B which needs affirmation. The projection V is meant to be an assumed theory and the possible states d of a base relation represent abducibles required to explain the changed view. Abductive reasoning now generates alternative explanations, i.e., possible changes t_u of the base relation, in order to match the changed view $u(V(d))$. This method can be useful as a heuristic to find a good explanation. However, since there are multiple views $V(t_u(d))$ as possible explanations, an exact solution would either require deductive reasoning in order to draw or derive the conclusion or an exhaustive inductive affirmation. Moreover, abductive reasoning is subject to the fallacy that an observation B is solely based on the order of events for which A is causally necessary and sufficient (*Post hoc ergo propter hoc*).

In summary, causation can neither be reduced to abductive reasoning nor deductive rules. In fact, user-view entanglement inseparably links user and DBMS and, unlike physically instantiated causal functions, there are no necessary and sufficient conditions for a view update to occur. According to the quantum interpretation presented in this article, failed attempts to deductively or abductively derive base relation updates from view updates have a reasonable explanation. Practically, there is not *one* solution to the view-update problem as traditionally conceived. It rather dissolutes if one accepts that external views are literally perspectives inseparably bound to their users.

5 Conclusion and Outlook

A quantum explanation of the indeterminate transformation $u \rightarrow t_u$ draws from the user's causation. The user cannot give necessary and sufficient conditions for a view update to occur. Accordingly, causation cannot be reduced to relational operators physically implemented as causal functions. User-view states are represented as entangled states ψ which require causation to become actual and thus determinate. Such nonseparable states ψ cannot be reduced to an updated view state $|1\rangle \in B_2$ for which there is a transformation $u \rightarrow t_u$ such that $u(V(d)) = V(t_u(d))$ or a state $|0\rangle \in B_2$ for which there are only transformations $u \rightarrow t_u$ such that $u(V(d)) \neq V(t_u(d))$. Therefore, there is a natural and reasonable explanation for the indeterminacy of $u \rightarrow t_u$ and thus for the view-update problem.

What conclusions can be drawn with regard to the ANSI-SPARC architecture and information systems in general? Generally, it must be acknowledged that persistent storage of data is the backbone of most application programs. Accordingly, user interfaces comply with the general idea of abstracting from the underlying physical data organisation as the ANSI-SPARC architecture illustrates for relational databases. However, and this is the main upshot of this article, one can argue that external views gain a new quality due to their inseparability from the user. The user-view whole, an indeterminate, irreducible and integrated set of states, poses exciting questions about the nature of such states. Causation, i.e., the user's effort to actively change his view and thereby the view as provided by the information system, transcends a clear distinction between user and view. The source of data for modelling user and view as an integrated and undifferentiated realm becomes less an analytical task of separation or decomposition. Rather it turns into a challenge of describing the synthesizing ways, or modes, such views change as a function of causation. There is much to learn from the cognitive sciences, an interdisciplinary field comprising subjects like psychology, artificial intelligence and philosophy. Here, recent theories conceptualize the user as an embodied agent whose perspectives change as a function of movements or motor habits. Taking the user as an embodied agent, who is integrated with external views of different representational formats, e.g., graphical, formal or textual, constitutes a rich and promising data source. The user's experiential episodes, carefully described from the background of his expertise in doing so, could possibly lead to a better understanding of the user's practical engagement with information technology. A paradigmatic example of practical reasoning is absorbed skilful coping. Here, users generally do not adopt a scientific attitude towards a subject matter, e.g., they do not analyse their engaged activity like they would construct a logical argument. People dealing with all sorts of artefacts usually immerse themselves in such a way that they gain a maximal grip on the contingencies in their environment. Simply presupposing a clear separation between user and computational de-

vice distorts the phenomenon of what it is actually like to deal with such a device. Although user-view entanglement is an undifferentiated realm for analysis this is not to say that practical reasoning is indifferent to synthesis. Several aspects of human-computer interactions, from simple viewings of pictures to deliberate reflections upon screen content, are hidden, or opaque, and therefore require special treatment. For instance, revealing aspects like the apprehension of a pictorial subject through perception of a pictorial image requires users to attend to the activity of picture viewing. Correlative to the actual content of a picture, picture viewing is a synthetic unity of perceptual and imaginative aspects. During the last century, several methods for describing the synthetic unity of such aspects have been worked out. They stem from the tradition of continental philosophers like Edmund Husserl, Martin Heidegger and Maurice Merleau-Ponty.

References

Aerts, D. & Gabora, L. (2005*a*), 'A State-Context-Property model of concepts and their combinations I: The structure of the sets of contexts and properties', *Kybernetes* **34(1&2)**, 167–191.

Aerts, D. & Gabora, L. (2005*b*), 'A State-Context-Property model of concepts and their combinations II: A Hilbert space representation', *Kybernetes* **34(1&2)**, 192–221.

ANSI-SPARC (1975), Data base management systems: Interim report., Technical report, FDT, ACM SIGMOD bulletin. Volume 7, No. 2.

Bancilhon, F. & Spyratos, N. (1981), 'Update semantics of relational views', *ACM Transactions on Database Systems* **6**(4), 557–575.

Bruza, P., Kitto, K., Nelson, D. & McEvoy, K. (2008), Entangling words and meaning, *in* 'Proceedings of the Second Quantum Interaction Symposium', University of Oxford.

Bruza, P., Lawless, W., van Rijsbergen, C., Sofge, D., Coecke, B. & Clark, S., eds (2008), *Proceedings of the Second Quantum Interaction Symposium*, College Publications, University of Oxford.

Bruza, P., Sofge, D., Lawless, W., van Rijsbergen, C. & Klusch, M., eds (2009), *Proceedings of the Third Quantum Interaction Symposium, University of Saarbrücken*, Vol. 5494 of *Lecture Notes in Artificial Intelligence*, Springer, Saarbrücken.

CODASYL (1971), Feature analysis of data base management systems, Technical report, ACM, New York.

Codd, E. (1970), 'A Relational Model of Data for Large Shared Data Banks', *Communications of the ACM* **13**(6), 377–387.

Codd, E. (1975), 'Recent investigations in relational database systems', *ACM Pacific* pp. 15–20.

Dayal, U. & Bernstein, P. (1982), 'On the correct translation of update operations on relational views', *ACM Transactions on Database Systems* **8**(3), 381–416.

Flender, C., Kitto, K. & Bruza, P. (2009), Beyond Ontology in Information Systems, *in* 'Proceedings of the 3rd International Symposium on Quantum Interaction', Springer, pp. 276–288.

Furtado, A. & Casanova, M. (1985), 'Updating relational views', *Query Processing in Database Systems* pp. 127–144.

Kakas, A. & Mancarella, P. (1990), Database updates through abduction, *in* 'Proceedings of the 16th International Conference on Very Large Databases', pp. 13–16.

Kitto, K. (2008), Why quantum theory?, *in* 'Proceedings of the Second Quantum Interaction Symposium', University of Oxford.

Lechtenbörger, J. & Vossen, G. (2003), 'On the computation of relational view complements', *ACM Transactions on Database Systems* **28**(2), 175–208.

Piwowarski, B. & Lalmas, M. (2009*a*), A Quantum-based Model for Interactive Information Retrieval, *in* 'Proceedings of the 2nd International Conference on Theory of Information Retrieval', Springer, pp. 232–240.

Piwowarski, B. & Lalmas, M. (2009*b*), Structured Information Retrieval and Quantum Theory, *in* 'Proceedings of the 3rd International Symposium on Quantum Interaction', Springer, pp. 289–298.