THE EFFECT OF CONFIDENTIALITY ON THE STRUCTURE OF DATABASES

Adrian Spalka and Armin B. Cremers

Abstract We give in this work an answer to the question of how real-life confidentiality can be introduced to an open database in a declarative manner. We present five forms of real-life confidentiality, translate them in the context of logic and show that only two forms can be brought in accord with databases that make the Closed World Assumption. These two forms, which correspond to real-life situations in which we do not want to admit that we are trying to keep something secret, can be precisely defined as distortions of the database's intended model. The main result of this work is a definition of a database with unequivocal static and dynamic semantics that supports real-life confidentiality at a declarative level. Two of its properties are that we never encounter polyinstantiation and that there is no inference.

Keywords: Databases, security, confidentiality, models, distortion

1. INTRODUCTION

Many works on databases commence with the statement that a database is generally regarded as an image of a given section of the real world. If we take a closer look at open relational and logic-based databases, we notice that great care has been taken to define their components in such a way that, within the chosen expressive frame, they represent the best possible copy of the elements of the real-world section. We cannot say that this is also true if we take a look at confidentiality. The only means a database system offers is a blurred and inadequate concept of rights. It is blurred because the right to select an element and the rights to insert, delete and update elements are grouped together. But the first right relates to confidentiality, which is the ability to get to know something, and the remaining rights relate to competence and powers, which is the ability to do something. And it is inadequate because access restriction is mere-

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: [10.1007/978-0-387-35508-5_22](https://doi.org/10.1007/978-0-387-35508-5_22)

V. Atluri et al. (eds.), *Research Advances in Database and Information Systems Security*

© IFIP International Federation for Information Processing 2000

ly an often necessary tool for achieving confidentiality but there are only a few situations in which it is also sufficient. In order to introduce confidentiality to a database as a sound and useful concept, we must first investigate its forms in the real life. We must determine what do we precisely mean and want to achieve when we say ‘*X* should be kept secret from *B*.’; we must investigate the ways we use to achieve this and the assumptions and preconditions on which the ways rely. And, lastly, we must find out which of the forms and ways can be expressed in and handled by a relational or logic-based database with the Closed World Assumption (CWA).

Our approach is based on the following observation. The intended state of an open real-world section (OWS) is reflected in the intended model of an open database (ODB). A user who has access to the ODB can see all of it and modify it according to his powers. A database with confidentiality demands (CDB) must therefore correspond to a real-world section with confidentiality demands (CWS). The introduction of confidentiality into the OWS implies that the intended state of the OWS is distorted in front of a user from whom an element of the OWS should be kept secret – the OWS still exists, but the user is shown a CWS, which is a distorted OWS. To imitate this situation in databases, we must distort the intended model of the ODB and show the user a CDB such that the intended model of the CDB is a distortion of the intended model of the ODB.

We show in this paper that there are three possible kinds of distortions: those based on external conventions, single-model and multi-model, and that these distortions can be derived from the ways confidentiality is handled in the daily life. We also identify invariants of the ODB which cannot be distorted for if we do it the result can no longer be called a database. And, lastly, we present some criteria for the satisfiability of confidentiality demands.

The first kind of distortions stems from the observation that if somebody is looking at something he does not know, instead of telling him what it really is we can tell him that it is something else. This kind does not require any changes to the database for we only distort the external convention of the meaning of some database elements.

Single-model distortions transform the ODB with its single model into a CDB with also only a single model. In real life, this process corresponds to situations in which we do not want to admit that we are trying to keep something secret. We present the user of the CDB with a single distorted model hoping that he is unable to notice the distorted parts of it. Single-model distortions are particularly difficult to realise since you must ensure that all invariants are satisfied and the users’ powers are not violated.

Lastly, multi-model distortions transform the ODB with its single model into a CDB with many models. The parallel to it in real life is a situation in which you admit that you have a secret or in which you pretend that you really

do not know the answer to a question, though you do. The problem with this kind is that, by definition, a database with the Closed World Assumption always has a single model¹. We also investigate the informal and formal sides of these distortions but we regard them as incompatible with our database.

In today's databases the static and dynamic semantics and integrity constraints have a clean declarative definition, whereas confidentiality is most often reduced to low-level operational access rules. This situation makes it hard, if not impossible, to conduct formal examinations and derive properties capable of proof with respect to confidentiality. The main contribution of this work is a declarative definition of confidentiality in databases with the CWA. The definition relies on the notion of a model which is a precise and well-understood element of mathematical logic. We also show that this is not a superficial definition – it reflects exactly two forms of confidentiality of the real life and, therefore, it is useful and easy to understand. It enables us to state confidentiality demands in the same declarative manner as, eg, we translate invariants of the real-world section into integrity constraints and we can formally investigate their satisfiability. The approach is undeniably more complex than a couple of access rules, yet our secrets in the daily, real life prove that we are well able to handle it.

The subsequent section presents a formal definition of a database. Section three focuses on previous works in this area. In section four we introduce some extensions to an open database which are necessary preliminaries to the introduction of confidentiality. The central part of this work, section five, discusses the effect of the various forms of confidentiality on the structure of databases. In the last section a conclusion briefly summarises our findings and presents a short outlook.

2. BASIC DEFINITIONS

An alphabet $\mathcal{A}$ is a non-empty and finite set of symbols, such that any string of $\mathcal{A}$'s elements has a unique decomposition into $\mathcal{A}$'s elements. $\mathcal{A}^*$ denotes the set of strings of finite length of elements of $\mathcal{A}$. Let $F = \mathcal{A}^*$ be the set of function symbols, $P \subseteq \mathcal{A}^*$ a finite set of predicate symbols, and $\rho_F : F \rightarrow \{0\}$ and $\rho : P \rightarrow \mathbb{N}_0$ the functions determining the arity of the symbols. Then $\Sigma = (\mathcal{A}, P, \rho)$ is a database signature. Let V be an infinite set of variables. Then $T(\Sigma) = F \cup V$ is the set of terms over the signature Σ . $T_G(\Sigma) = F$ is the subset of ground terms. The set of atomic formulae over Σ is $A(\Sigma) = \{p(t_1, \dots, t_n) \mid p \in P, \rho(p) = n, t_i \in T(\Sigma), i = 1, \dots, n\}$ and

¹ If the application of the CWA to a database renders it inconsistent or yields several models, then the CWA is regarded as incompatible to the database and is not applied to it.

$A_G(\Sigma)$ is the subset of ground atomic formulae. With $\alpha \in A(\Sigma)$, α is a positive literal and $\neg\alpha$ a negative literal. The first order language over the signature Σ is the smallest set $L(\Sigma)$ with the following properties: $A(\Sigma) \subseteq L(\Sigma)$; if $\varphi, \psi \in L(\Sigma)$, then $(\varphi) \vee (\psi) \in L(\Sigma)$; if $\varphi \in L(\Sigma)$, then $\neg(\varphi) \in L(\Sigma)$; if $\varphi \in L(\Sigma)$ and $X \in V$, then $\forall X : \varphi \in L(\Sigma)$. $L_0(\Sigma)$ is the subset of closed formulae, viz, all variables of a $\varphi \in L_0(\Sigma)$ are quantified. A normal clause is a closed formula $\alpha \leftarrow \lambda_1 \wedge \dots \wedge \lambda_n$ in which all variables are assumed to be universally quantified, $\alpha \in A(\Sigma)$ is the clause's head and $\lambda_1 \wedge \dots \wedge \lambda_n$, a conjunction of literals, its body. A clause is range-restricted if any variable that occurs in the clause occurs also in a positive literal in its body. $C_R(\Sigma) \subseteq L_0(\Sigma)$ is the set of range-restricted clauses over Σ .

Let $\mathfrak{D}$ and Ω be sets such that there is a $\omega_F \in \Omega$, $\omega_F : F \rightarrow \mathfrak{D}$ and for each $p \in P$, $\rho(p) = n$, there is a $\omega_p \in \Omega$, $\omega_p : \mathfrak{D}^n \rightarrow \{\text{True}, \text{False}\}$. Then $\hat{M}(\Sigma) = (\mathfrak{D}, \Omega)$ is an interpretation for the signature Σ . Alternative notations for restrictions to $\hat{M}(\Sigma) = (\mathfrak{D}, \Omega)$:

- for ground atomic formulae: $M(\Sigma) : A_G(\Sigma) \rightarrow \{\text{True}, \text{False}\}$
- for closed formulae: $M_0(\Sigma) : L_0(\Sigma) \rightarrow \{\text{True}, \text{False}\}$.

$\hat{M}(\Sigma)$ is a model of Φ , $\hat{M}(\Sigma) \in \text{Mod}(\Phi)$, $\Phi \subseteq L_0(\Sigma)$, if $\forall \varphi \in \Phi : M_0(\Sigma)(\varphi) = \text{True}$. (Some works on databases define a model as $M^{-1}(\Sigma)(\text{True}) \subseteq A_G(\Sigma)$.) $\hat{M}(\Sigma) = (\mathfrak{D}, \Omega)$ is a Herbrand-interpretation or a Herbrand-model if $\mathfrak{D} = T_G(\Sigma)$ and $\omega_F = \text{id}$. $\Psi \subseteq L_0(\Sigma)$ is a logical consequence of $\Phi \subseteq L_0(\Sigma)$, $\Phi \models \Psi$, if $\text{Mod}(\Phi) \subseteq \text{Mod}(\Psi)$.

We assume that the intended semantics of a set $\Phi \subseteq L_0(\Sigma)$ is defined by its completion² with respect to Σ , $\text{comp}(\Phi, \Sigma)$.

Let Σ be a database signature, $C \subseteq L_0(\Sigma)$, $\text{Mod}(C) \neq \emptyset$, a finite set of closed formulae over Σ and $I \subseteq C_R(\Sigma)$ a finite set of range-restricted clauses over Σ such that $\text{comp}(I, \Sigma) = C$.³ Then $D = (\Sigma, C, I)$ is a logic-based database with completion semantics. Σ defines the database language, C is the set of integrity constraints, I is a valid present state and the intended semantics of I is defined by the unique Herbrand-model $M(\Sigma)$ of $\text{comp}(I, \Sigma)$.

3. PREVIOUS AND RELATED WORK

In SQL only the GRANT/REVOKE SELECT statements, which apply only to relations, support confidentiality. Formally, the open database $D = (\Sigma, C, I)$ with $\Sigma = (\mathfrak{A}, P, \rho)$ has a set P of relations. If a user u does not possess the SELECT right to some relations, then there is a database

² Cf, eg, [2].

³ This definition ensures that the integrity constraints are satisfiable and that the completion has a unique Herbrand-model.

$D_u = (\Sigma_u, C_u, I_u)$ with $\Sigma_u = (\mathcal{Q}, P_u, \rho_u)$ such that $P_u \subseteq P$ and the remaining components of D_u are restrictions of the components of D to the elements of P_u . Informally, we use this method to simply lie to the user: if an excluded relation is a base relation, then we lie to him about the extent of the real-world section which is known to the database and if the excluded relation is a view with an obvious connection to a base relation, then we lie to him about the truth value of the withheld tuples (they are true in the intended model of I but false in that of I_u). Note that D and D_u are not two independent databases. D_u is a distorted copy of D , which corresponds to a distorted image of the real-world section and any change applied to D_u is immediately propagated to D . The SQL-approach has two disadvantages. Firstly, there is no way of storing in D_u tuples that are not in D , ie, we cannot set a tuple's truth value to False in D and to True in D_u . And, secondly, we have not found any manual, which tells you how to define P_u in such a way that the restriction of D to D_u results in a valid database D_u and that a transaction accepted by D_u will not be rejected by D , ie, that D_u retains the static and dynamic semantics of a database.

Most other algebraic approaches are based on SeaView, a multi-level relational data model introduced by [3]. It assumes that there is an open database D which is associated with the highest security level of a lattice of security levels and that a database D_l is associated with any other security level l . Since each D_l is a restriction of D , the approach is in fact similar to that of SQL. The only difference is that in addition to reducing the set of relations, SeaView can also reduce the state. Thus, SeaView lies to the user in the same way as SQL does. But whereas SQL provides a clear informal concept and formal definition of its notion of confidentiality, SeaView does not. The idea to bring NULL-values in connection with confidentiality and the idea that polyinstantiation – a phenomenon invented to justify a flawed design – is natural and inevitable resulted in situations in which the meaning of confidentiality is unclear. The obvious consequence of these facts is that SeaView inherits all disadvantages of SQL and adds to it a pile of its own problems.

There are several works, eg [8], [5] and [1], which regard a formula as a unit of protection and a formula's confidentiality is defined as non-derivability. Whereas derivability is a unique notion, non-derivability can be formalised in three ways. Suppose that a formula $\varphi \in L(\Sigma)$ should be kept secret from the user u . Then, firstly, φ is not derivable in D_u if φ cannot be expressed in the language of D_u , ie, if $\varphi \notin L(\Sigma_u)$. Or, secondly, if φ is in D , then it is not in D_u or vice versa, ie, if the truth value of φ in D_u is opposite to that in D , ie $M(\Sigma)(\varphi) = \neg M_u(\Sigma_u)(\varphi)$. Or, lastly, if all true statements, which comprise φ , can be only reduced to a non-singleton OR-statement, ie, to a formula $\varphi \vee \psi$ with $\varphi \neq \psi$. This situation implies that D_u does not have a unique intended model but many and there are two models M_u^i and M_u^j such that $M_u^i(\Sigma_u)(\varphi) = \neg M_u^j(\Sigma_u)(\varphi)$. This last interpretation is chosen by [8] and [1];

[5] does not tackle the choices.

Based on the conviction that standard logic is inadequate, [10] attempts to formalise the rules and properties of mandatory access control models⁴ in NTML, a non-monotonic logic, which, however, has been shown to be not sound. The main step in NTML is the assignment of a security level to all elements of the signature and to all formulae of D . $\Sigma_u = (\mathcal{Q}_u, P_u, \rho_u)$ is determined by the user's current security level and the components of D_u are restrictions of the components of D to Σ_u .

4. USERS AND POWERS IN OPEN DATABASES

Before we can investigate the forms of confidentiality, we must realise that the ability to keep a unit of protection secret from a user is limited by two fundamental factors: the user's powers to change the present state, which must be respected, and the user's ability to learn about the unit's current state or existence without asking the database. Therefore, the first extension to an open database with respect to confidentiality is the inclusion of these concepts.

Let $D = (\Sigma, C, I)$ be an open database. We assume that:

- there is a finite set $U = \{u_1, \dots, u_n\}$ of users with legitimate access to D
- for each $u \in U$ there is a set $R_u \subseteq A_G(\Sigma)$ that reflects the user's powers in the real-world section, which means that a transaction T with the effect τ executed by u
 - is rejected if τ is not a subset of R_u , and
 - must be accepted by D if τ is a subset of R_u and does not violate the integrity constraints

We say that T is authorised if τ is a subset of R_u .

- for each $u \in U$ there is a set $K_u \subseteq A_G(\Sigma)$ that reflects the user's specific knowledge on the real-world section, ie his ability to learn about the current state or existence of some units without asking the database

Then $D = (\Sigma, C, I, U, \{R_u\}_{u \in U}, \{K_u\}_{u \in U})$ is an open extended database.⁵

The rights R_u determine the transactions user u is authorised to execute. While the decision on a transaction's validity is made by the integrity constraints, an organisation's or an application's security policy is solely responsible for the granting and the revocation of an authorisation. This means that we must distinguish the reasons because of which a transaction can be rejected. If it is because of a violation of the state's validity, then the integrity constraints

⁴ Cf, eg, [4].

⁵ Note that there are no confidentiality demands, ie, the users are still allowed to know everything.

make the reasons explicit, and, at the same time, provide the justification – both are visible and comprehensible to the user. If it is for lack of authorisation, then the user can neither find the reasons nor the justification within the database. The security policy is only reflected in the rights – it is not stated there. Thus if a user is puzzled about a missing right, then it is not the database, but the persons who set up the security policy that are responsible for providing a satisfactory explanation. Since this work is on confidentiality, the consequences of failing to do so can be neglected from our viewpoint as long as no information must be kept secret from the user. Otherwise, this behaviour may well affect a user's trustworthiness in the negative.

The purpose of the set K_u is that its elements cannot be kept secret from u . The question of how to determine K_u precisely has, of course, no single true answer. But the semantics of the powers, R_u , tells us that the least a user can know about the real-world section is what he has arranged or is able to do so, ie, R_u is a lower bound on K_u .

ASSUMPTION 1 $R_u \subseteq K_u$ always holds.

5. CONFIDENTIALITY IN DATABASES

This chapter investigates the transition from an open database (ODB), which corresponds to an open real-world section, to a database with confidentiality demands (CDB), which corresponds to a real-world section with confidentiality demands.

5.1 Distortion of the intended external convention

The algebraic and logic-based theory of databases is not concerned with the question of how does a database user know the real-life meaning of the relation names and the attribute values, ie the elements of the signature. As long as the database is open, this question is, admittedly, of little avail to the theories. However, the situation changes as soon as we introduce confidentiality.

The actual correspondence between the elements of the real-world section and the elements of the signature is established in two steps. Firstly, the persons who communicate with the real-world section are identified with the users who communicate with the database. Secondly, the real-world meaning of an element of the signature is fixed in the intended external convention, a convention intelligible and known to all legitimate users.

In open databases the real-world objects and their properties are taken to be indistinguishable from their names in the signature of which the users have an implicit understanding. One could object that if this level of abstraction was questioned, then one could proceed ad infinitum, and the ironic end would be

the situation in which one is not able to refer to a real-world object in a conversation but to point with a finger to this object itself. In practice, however, this argument misses the nature of codes, watchwords and other secret arrangements. Their usefulness relies on the fact that a group of people is in the know of only a single additional level of indirection with respect to some names' natural-language semantics. The sentence 'Tonight we go to the pub' gains a different meaning for the members of a gang who, in order to hide their intentions, have agreed to denote a bank as a pub. To take this situation into account, we assume that a signature is a syntactical object with semantic intentions in respect of its real-world meaning, which are stated in an explicit convention.

Let $\mathbb{C}$ be the intended external convention of $\Sigma = (\mathcal{A}, P, \rho)$, the signature⁶. If an element of Σ should be kept secret from a user u , we can present to him $\mathbb{C}_u$, a distortion of $\mathbb{C}$.

EXAMPLE 1 Suppose that

- `hot_fields` is a relation, ie an element of P , and its intended external convention is $\mathbb{C}(\text{hot_fields}) = \text{'is a secret airfield'}$
- $\text{red} \in T_G(\Sigma)$ is an attribute value and $\mathbb{C}(\text{red}) = \text{'alert condition red'}$

If a user is unaware of the intended meaning of these symbols and we would like to keep it secret from him, then we can offer him the following external convention $\mathbb{C}_u$, which is a distortion of $\mathbb{C}$:

- $\mathbb{C}_u(\text{hot_fields}) = \text{'is a favourite holiday spot'}$
- $\mathbb{C}_u(\text{red}) = \text{'the colour red'}$ $\bowtie$

First of all, we do not gain any relevant insight if we try to formalise $\mathbb{C}$ since it is the final level of describing the meaning of an image of a real-world section. For that reason the database has no way of influencing the intended external convention. On the one hand, this makes this distortion quite simple to apply because the database does not need to be changed in any way. On the other, its applicability is limited to only a few situations. Assumption 1 tells us that the user has no update rights to tuples with a distorted convention. Otherwise, we were in for a home-made sabotage. Suppose that user u of Example 1 changes `red` to `green` because he wants to give something a fresh look – this is a, possibly unintended, distortion of the open database with respect to the intended external convention and it confuses all users who rely on it. It is also useless in many other situations. It may be only of some help if the user is a guest who somehow sees a part of the database.

Since distortions of the intended external convention cannot be influenced by the database we do not examine it further and assume from now on that all users $u \in U$ have a uniform intended external convention.

⁶ Note that a convention does not apply to the tuples' truth values. All elements of the database state are taken as true and all remaining ones as false.

5.2 Multi-model distortions

Multi-model distortions are a formalisation of two situations:

- i) somebody pretends not to know the answer to a question though he does
- ii) somebody admits to have a secret

The intention in both cases is to construct a CWS by introducing a certain degree of blur into the OWS.

An example of the first case is the question ‘Where does this plane go to?’ and the answer to it ‘I really do not know’. Since you know or see that the plane has left, this answer tells you that the plane may go to any airport within its reach and on which it can land. Suppose that the meaning of the predicate W is ‘is the destination of this plane’, then, formally, this answer can be represented as the finite disjunction $W(\text{Rome}) \vee W(\text{Paris}) \vee \dots \vee W(\text{London})$, if Rome, Paris and London qualify as possible destinations. We can describe this situation with respect to the open database with its unique intended model $M(\Sigma)$ as follows. Suppose that Rome is the intended destination. Then the CDB for the user u , who asks the question, is constructed by replacing $M(\Sigma)$ with a finite set of models $\mathfrak{M}_u(\Sigma)$ such that:

- i) The CDB claims that $\mathfrak{M}_u(\Sigma)$ is the smallest set of its models
- ii) $|\mathfrak{M}_u(\Sigma)| > 1$, ie the state of some elements of the real-world section is unknown and there is no unique intended model
- iii) $M(\Sigma) \in \mathfrak{M}_u(\Sigma)$, ie the intended destination is among the possible destinations
- iv) There is at least one $M_u^i(\Sigma) \in \mathfrak{M}_u(\Sigma)$ such that $M_u^i(\Sigma)(W(\text{Rome})) = \text{False}$, ie at least one choice tells you that Rome is not the intended destination

An example of the second case is the question ‘How much do you earn?’ and the answer to it ‘I know it but I will not tell it’. Informally, you are told that the amount of the salary is secret. Formally, the answer can still be represented as a finite disjunction since the lowest amount is fixed by law and the highest amount by common sense. The formal representation of this case in the database is identical to that of the first case with the exception of points (i) and (ii):

- i) There is an ODB with a unique intended model and the CDB is a distortion thereof
- ii) $|\mathfrak{M}_u(\Sigma)| > 1$, ie the state of some elements of the real-world section is deliberately blurred

Of these two cases the first one is not applicable to our database because it violates its definition. The statement that the OWS cannot be represented more precisely than by a non-singleton set of models, $\mathfrak{M}_u(\Sigma)$, contradicts the CWA, which, if compatible to the database, always yields a unique intended model.

The second case has several problems. Firstly, in order to give an indefinite

answer, the semantics of the database has to be extended. Secondly, there are no satisfactory approaches to the representation of $\mathfrak{M}_u(\Sigma)$ and its operational treatment. Whereas these problems can be solved, its main flaw is shown by [8]. They use a censor who takes into account the user-knowledge and the dialogue up to that time and who decides on the database's answer. The censor, who never lies, gives the indefinite answer if he believes that the user will be able to deduct the secret with the help of a precise yes/no-answer. The authors demonstrate that this can still be of little avail since the user can make use of the censor's reasoning in his own one and discover the secret.

Thus, we conclude, multi-model distortions cannot be used to introduce confidentiality to an open database.

5.3 Single-model distortions

Let us now turn our attention to situations in which confidentiality is understood in the sense that we would also like to keep the fact confidential that we have a secret.

5.3.1 Distortion of the truth values

An example of such a situation is the question 'Where does this plane go to?' and the answer to it 'To London', though you know that Rome is its intended destination. Since you also want to keep it secret that your answer is a lie you must take many additional precautions. Firstly, you must ensure that your answer complies with the general knowledge on the real-world section, eg, that this plane can indeed fly to and land in London. Secondly, you must ensure that the person who asks the question is unable to modify your secret and any information related to it, eg, that he cannot order this plane to Paris or add a cargo of grapes, which never arrives in London. And, lastly, when confidentiality is no longer desired you must inform this person of its true destination, eg Rome, and remove the lie used to cover it, eg London.

This situation has the following formal correspondence. Suppose that the user u asks the question. Then you present to u a CWS with the following properties:

- You pretend in front of u that the statement 'This plane flies to Rome' has the opposite truth value in the CWS as in the OWS
- In order to satisfy the integrity constraint 'Every plane has a destination' you use an alias, London, and pretend in front of u that the statement 'This plane flies to London' has the opposite truth value in the CWS as in the OWS
- You must check that u does not have the powers to alter the plane's destination and cargo

- You must record that the statement ‘This plane flies to Rome’ should be kept secret in front of u and that the statement ‘This plane flies to London’ is only used as an alias for the first statement, so that when the truth value of the first statement in the CWS is adjusted to that in the OWS, the same must be done for the alias.

With this in mind, it is easy to translate the situation in the context of our database. Recall that

$$M(\Sigma) : A_G(\Sigma) \rightarrow \{\text{True}, \text{False}\} \quad (1)$$

is the intended model of the ODB. We now construct $D_u = (\Sigma_u, C_u, I_u)$, the CDB, with $M_u(\Sigma_u) : A_G(\Sigma_u) \rightarrow \{\text{True}, \text{False}\}$. Let α be the tuple which should be kept secret. Then:

- set $M_u(\Sigma_u)(\alpha) = \neg M(\Sigma)(\alpha)$, ie set the truth value of α in the CDB to the opposite of its truth value in the ODB
- if there are some integrity constraints $\psi_1, \dots, \psi_m \in C_u$ which are no longer satisfied due to step (i), try to find one or more aliases $\beta_1, \dots, \beta_n$ such that $\psi_1, \dots, \psi_m$ are satisfied if we set $M_u(\Sigma_u)(\beta_i) = \neg M(\Sigma)(\beta_i)$ for all $i = 1, \dots, n$
- check that $\alpha, \beta_1, \dots, \beta_n \notin R_u$ to ensure that u cannot destroy our secret
- if u can now execute a valid and authorised transaction with respect to D_u which, however, will be rejected by D , prevent it – try to find one or more aliases $\gamma_1, \dots, \gamma_k$ such that, if we set $M_u(\Sigma_u)(\gamma_i) = \neg M(\Sigma)(\gamma_i)$ for all $i = 1, \dots, k$, then every valid and authorised transaction with respect to D_u is also accepted by D
- to record all these distortions we must introduce an alias-log P :
 - insert $P(D_u, \alpha, \text{secret}, \{\beta_1, \dots, \beta_n, \gamma_1, \dots, \gamma_k\})$ into P
 - if some of the aliases $\delta_1, \dots, \delta_l \in \{\beta_1, \dots, \beta_n, \gamma_1, \dots, \gamma_k\}$ have required further aliases for themselves, then insert $P(D_u, \delta_i, \text{alias}, \{\delta_1^i, \dots, \delta_{r_i}^i\})$ into P for all $i = 1, \dots, l$

There are no changes to the signature Σ_u and to the integrity constraints C_u , ie $\Sigma_u = \Sigma$, $C_u = C$ and $D_u = (\Sigma, C, I_u)$. As long as we have a relational database, there is a simple correspondence between the content of the state I_u and the intended model $M_u(\Sigma)$.

Thus, we have constructed a function which represents the intended model of the CDB D_u ,

$$M_u(\Sigma) : A_G(\Sigma) \rightarrow \{\text{True}, \text{False}\} \quad (2)$$

which assigns to all tuples that do not occur in the alias-log P the same truth value as (1), the intended model of the CDB, does and which assigns to all tuples that occur in P , ie, which should be kept secret from u or are a conse-

quence of this confidentiality demand, the opposite truth value than (1) does. In this sense we can say that (2) is a distortion of (1) and the parameter of the distortion is the assignment of truth values.

To draw a parallel to real life, we hide a secret by telling some more lies and pretend that everything is in perfect order – at the same time we are well aware of the possibility that other people behave in the same manner in front of us.

This type of confidentiality applies only to ground atomic formulae, ie tuples – we cannot change the truth value of a term, ie an attribute value, or a relation because they, simply, do not have truth values. To include this type of confidentiality in our database we only have to include the alias-log P into it. Thus, the open extended database with users and powers

$$D = (\Sigma, C, I, U, \{R_u\}_{u \in U}, \{K_u\}_{u \in U})$$

now becomes a database with confidentiality

$$D = (\Sigma, C, I, U, \{R_u\}_{u \in U}, \{K_u\}_{u \in U}, P) \quad (3)$$

To construct a particular distorted database D_u we take the model of D , apply the distortions recorded in the alias-log P to it and then translate the distortions of the model to distortions of the state of D_u .

5.3.2 Distortion of the domain

An apparent shortcoming of the above-defined type of confidentiality is that we are unable to keep the existence of, eg, the starship Enterprise secret. Suppose that in the first step we remove the tuple Starships(Enterprise) from the Starships relation in the distorted database. But then, in step four, we notice that we need an alias to preserve the database semantics but there is none. Well, one can object that we can remove all tuples that comprise the name Enterprise from the user's update rights. But if he has the powers to buy new starships how do we justify in front of him the fact that he is not allowed to name a new starship Enterprise? This problem is quite important in practice since, eg, there may be a public hangar for the Voyager starship and a secret hangar for the Enterprise. To distinguish this type of confidentiality from the previous one, we must note that the object of confidentiality of the previous type is the truth value of a statement. Here, the object of confidentiality is the existence of objects and properties.

A solution to this problem, structured name-spaces, is presented in [9]. Here we show only how structured name-spaces can be interpreted as a distortion of the intended model.

We recall that real-world objects correspond to attribute values of a tuple or terms and properties to relations or predicates. In order to keep the existence of an object or a relation secret the database must pretend that it is not an element of the real-world section, ie, when asked about it a possible answer is ‘I do not know what you are talking about’. If we again take a look at the intended model, $M(\Sigma) : A_G(\Sigma) \rightarrow \{\text{True}, \text{False}\}$, then we note that the previous type of confidentiality modifies the value of this function at certain points. Now, to pretend that some objects or properties are unknown, this function must become undefined at all points, which comprise the secret object or property. The obvious way to achieve it is to remove these points from the function’s domain $A_G(\Sigma)$. Since $A_G(\Sigma)$ is determined by the signature Σ , we can modify it only by modifying Σ . Some of the previous works have shown that it is quite easy to present to a user u a database with a distorted signature $\Sigma_u = (\mathcal{Q}, P_u, \rho_u)$ from which some relations are removed, ie $P_u \subseteq P$. In this case the user is not told that, eg, there is a relation that stores a cargo’s value, but he can still ask about Enterprise. To prevent him from doing this we must structure the database’s flat name-space. To give an example, a file system can store two files with same name if they are in different directories. Here, the full path-name of the directory serves as a name-space selector and the name of the file is a local name within the selected name-space.

[9] come to the conclusion that following modifications to $\Sigma = (\mathcal{Q}, P, \rho)$, the signature, are necessary to obtain a database with structured name-spaces:

- $\mathcal{Q}^*$ is regarded as a universal name-space
- There is a new set E of name-space selectors
- $N = E \times \mathcal{Q}^*$ is the new global universal name-space and $(a, b) \in E \times \mathcal{Q}^*$ an element of it
- The sets of terms and predicates are now defined over N (and not over $\mathcal{Q}^*$)
- For each user $u \in U$ there is a set $E_u \subseteq E$ of name-space selectors available to u , which defines his personal universal name-space

$$N_u = \bigcup_{e \in E_u} \{e\} \times \mathcal{Q}^*$$

At a first glance, we must extend our database with two components: E and the set of all E_u . However, [9] show that if confidentiality is the purpose for using structured name-spaces, then setting

$$E = \wp(U) \tag{4}$$

the power-set of U , already offers maximum flexibility.⁷ We therefore assume (4) and give the final definition of a database with confidentiality:

⁷ It is also sufficient to model multi-level structures.

$$D = (\Sigma, C, I, U, \{R_u\}_{u \in U}, \{K_u\}_{u \in U}, P, \{E_u\}_{u \in U}) \quad (5)$$

We just need one more word on P . The intended model of D is now the function

$$M(\Sigma) : A_G(\Sigma) \rightarrow \{\text{True}, \text{False}\} \quad (6)$$

such that $A_G(\Sigma)$ is defined over N , and the intended model of a distorted database D_u is now the function

$$M(\Sigma_u) : A_G(\Sigma_u) \rightarrow \{\text{True}, \text{False}\} \quad (7)$$

such that $A_G(\Sigma_u)$ is defined over N_u . If we begin the construction of (7) by reducing (6) to $A_G(\Sigma_u)$, then we need to store in the alias-log P only the reversed truth values with respect to $A_G(\Sigma_u)$.

5.3.3 Some remarks

In (5) we have arrived at a database that supports two forms of confidentiality. It is a well defined database in the following sense:

- i) Its static semantics is identical to that of an open database, viz, both the open database and any distortion thereof have a unique intended model
- ii) Its dynamic semantics is a natural restriction of that of an open database since a transaction must be both valid and authorised to be accepted⁸
- iii) It defines the meaning of confidentiality in a precise declarative manner and there is a clear correspondence to the real life
- iv) As in real life it rejects confidentiality demands that are recognised to be not satisfiable

A consequence of point (i) is that we do not even need to think of polyinstantiation. Since we have a clear distinction between the ODB and any CDB, we always precisely know the intended state of affairs in the OWS and we have full control of the distortions in any CWS.

A consequence of the points (i), (ii) and (iv) is that we also do not need to think of inference. With respect to our approach, inference can be defined as the ability of a user of a CDB to discover one or more distortions and, thus, infer its intended state in the ODB. However, the way we have constructed (5) precludes exactly this possibility. Well, of course, the user of the CDB may know somebody who knows something about a distorted element – that's life, but what is more important is that our database cannot do anything about it. It

⁸ However, it is much harder to define and compute the effect of a transaction if the users are embedded in a hierarchy of groups. We will soon report on this problem in a paper.

can take into consideration only those elements of which it knows – and these are precisely stated in (5).

We have identified five forms of confidentiality and shown that one of it cannot be influenced by the database, two of them are not compatible with the CWA and the final two forms can be integrated in a database. There is probably no formal proof that there are no other forms of confidentiality but if we identify the semantics of a database with its intended model, then there are no other ways of distorting a function. The only additional distortion we can apply is to include alias-predicates in the distorted domain, ie, the CDB comprises relations that do not exist. However, we do not regard this distortion as a form of confidentiality but only as a tool for the satisfaction of confidentiality demands.⁹

Lastly, this paper gives only a definition of a database. The important question of how we determine whether a confidentiality demand is satisfiable and, if it is, how we compute the necessary distortions, is left open. We will present an answer to it in the near future.

6. CONCLUSION

The motivation for this work is the question of how we can introduce real-life confidentiality in an open database in a declarative manner. In a first step we have introduced users and their powers in the open database and argued that the powers users have in real life must also be respected by the database, which implies that there is a set of clearly not satisfiable confidentiality demands. We have then presented five forms of real-life confidentiality, we have translated them in the context of logic and shown that only two forms can be brought in accord with the Closed World Assumption in our database. These two forms, which correspond to real-life situations in which we do not want to admit that we are trying to keep something secret, can be precisely defined as single-model distortions of the database's intended model. Our view that confidentiality must be seen as a distortion of an open world-section has led us to a database with confidentiality, which is an extension to open databases, ie, it possesses unequivocal static and dynamic semantics. A consequence of this and some other properties is that we never encounter polyinstantiation and that there is no inference. The most important result of this work is the final definition of a database with confidentiality. We have shown that the components of this database are necessary in order to introduce real-life confidentiality to an open database.

⁹ We presently investigate whether there are situations in which the inclusion of false predicates is necessary.

In the near future we will present more details on the definition and execution of transactions in our database and some methods for the enforcement of confidentiality demands.

Lastly, we would like to make a remark on a problem that is closely related to confidentiality, though not an issue of the database design. In a database that supports confidentiality, users and programs can receive incomplete or wrong information. This distorted information is often used in further decisions. We believe that at this stage it is the responsibility of users who state confidentiality demands to contemplate on their negative consequences and implications.

References

- [1] Bonatti, P., Kraus, S. and Subrahmanian, V.S. (1992). Declarative foundations of secure deductive databases. Ed J. Biskup and R. Hull. *4th International Conference on Database Theory – ICDT'92*. LNCS, vol 646. Berlin: Springer-Verlag. pp. 391-406.
- [2] Cremers, A. B., Griefahn, U. and Hinze, R. (1994). *Deduktive Datenbanken*. Braunschweig: Vieweg.
- [3] Denning, D. E., Lunt, T. F., Schell, R. R., Heckman, M. and Shockley, W. R. (1987). A multilevel relational data model. *1987 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press. pp. 220-234.
- [4] Landwehr, C. E. (1981). Formal models for computer security. *ACM Computing Surveys* **13**(3), pp. 247-278.
- [5] Morgenstern, M. (1987). Security and inference in multilevel database and knowledge-base systems. *1987 ACM SIGMOD Conference / SIGMOD Record* **16**(3), pp. 357-373.
- [6] Morgenstern, M. (1988). Controlling logical inference in multilevel database systems. *1988 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press. pp. 245-255.
- [7] Reiter, R. (1984). Towards a logical reconstruction of relational database theory. Ed M. L. Brodie, J. Mylopoulos and J. W. Schmidt. *On Conceptual Modeling*. New York: Springer-Verlag. pp. 191-238.
- [8] Sicherman, G. L., de Jonge, W., and van de Riet, R. P. (1983). Answering queries without revealing secrets. *ACM Transactions on Database Systems* **8**(1), pp. 41-59.
- [9] Spalka, A. and Cremers, A. B. (1997). Structured name-spaces in secure databases. Ed T. Y. Lin and S. Qian. *Database Security XI*. IFIP TC11 WG11.3 Conference on Database Security. London at al: Chapman & Hall, 1998. pp. 291-306.
- [10] Thuraisingham, B. M. (1991+1992). A nonmonotonic typed multilevel logic for multilevel secure data/knowledge base management systems I+II. *The Computer Security Foundations Workshop IV+V*. IEEE Computer Society Press. pp. 127-138/135-146.