applied sciences

Communication

Ensuring Data Integrity in Databases with the Universal Basis of Relations

Vitalii Yesin [1], Mikolaj Karpinski [2,*], Maryna Yesina [1,*], Vladyslav Vilihura [1] and Kornel Warwas [2]

[1] Department of Security of Information Systems and Technologies, Faculty of Computer Science, V. Karazin National University of Kharkiv, 61022 Kharkiv, Ukraine; v.i.yesin@karazin.ua (V.Y.); viligura93@gmail.com (V.V.)
[2] Department of Computer Science and Automatics, Faculty of Mechanical Engineering and Computer Science, University of Bielsko-Biala, 43-309 Bielsko-Biala, Poland; kwarwas@ath.bielsko.pl
* Correspondence: mkarpinski@ath.bielsko.pl (M.K.); m.v.yesina@karazin.ua (M.Y.)

Abstract: The objective of the paper was to reveal the main techniques and means of ensuring the integrity of data and persistent stored database modules implemented in accordance with the recommendations of the Clark–Wilson model as a methodological basis for building a system that ensures integrity. The considered database was built according to the schema with the universal basis of relations. The mechanisms developed in the process of researching the problem of ensuring the integrity of the data and programs of such a database were based on the provisions of the relational database theory, the Row Level Security technology, the potential of the modern blockchain model, and the capabilities of the database management system on the platform of which databases with the universal basis of relations are implemented. The implementation of the proposed techniques and means, controlling the integrity of the database of stored elements, prevents their unauthorized modification by authorized subjects and hinders the introduction of changes by unauthorized subjects. As a result, the stored data and programs remain correct, unaltered, undistorted, and preserved. This means that databases built based on a schema with the universal basis of relations and supported by such mechanisms are protected in terms of integrity.

Keywords: integrity; database; database with the universal basis of relations; Clark–Wilson model

check for updates

Citation: Yesin, V.; Karpinski, M.; Yesina, M.; Vilihura, V.; Warwas, K. Ensuring Data Integrity in Databases with the Universal Basis of Relations. *Appl. Sci.* **2021**, *11*, 8781. https://doi.org/10.3390/app11188781

Academic Editor: Gianluca Lax

Received: 29 August 2021
Accepted: 19 September 2021
Published: 21 September 2021

1. Introduction

Ensuring information security of databases (DBs) is impossible without considering aspects of ensuring data integrity. Many, especially commercial, organizations are more concerned with the integrity of their data than its confidentiality [1]. Integrity is more important to them. If you publish information on the Internet on a web server and your goal is to make it available to the widest possible range of people, then confidentiality is not required. On the contrary, the responsibility for providing undistorted information obtained from a database, for example, about the data stored in it from official legal, regulatory, financial, medical, and other documents of the organization, including these documents themselves, is significantly increased. The information must be authentic or genuine. Data must remain correct, truthful, and be a true reflection of reality. In general, both in a commercial and a military environment, it is difficult to imagine a system for which the properties of integrity would not be important [2].

As noted in the Certified Information Systems Security Professional Official Study Guide [1], numerous attacks are aimed at violating integrity. These are both malicious modifications performed by various malicious programs and errors in applications. Integrity violations are not limited to deliberate attacks. User error, oversight, or inept actions are the cause of many cases of unauthorized modifications of information. Events that lead to integrity violations include the modification or deletion of files, database data, entry of incorrect data, configuration alteration, errors in commands, virus introduction, and malicious code execution. Integrity violations can occur due to the actions of any user, including administrators, either through an oversight in the security policy or due to misconfigured security controls.

The authors of the information systems security guide [1] noted that integrity can be examined from three perspectives: Preventing unauthorized subjects from making modifications, preventing authorized subjects from making unauthorized modifications (e.g., errors), and maintaining internal and external consistency of objects. Properly implemented integrity protection provides a means for authorized modifications while protecting against malicious unauthorized actions, as well as errors made by authorized users. This ensures that the data remain correct (there are no logical errors in the structure and data values), unaltered (data identity to a certain standard), undistorted (no data tampering), and preserved. When a security mechanism ensures integrity, it provides a high level of assurance that data, objects, and resources will not be altered from their original protected state. However, at the same time, it should be remembered and taken into account that integrity control requires additional resources: Time and memory. For example, the main problem in the implementation of mechanisms for controlling the integrity of file objects is their rather strong influence on the load of the computing resource of the system, which is due to the following reasons [3]: First, control of large amounts of information may be required, which is associated with a significant duration of the control procedure; second, continuous maintenance of the object in a reference state may be required. In this connection, a natural question arises: With what frequency to exercise control, since file integrity monitoring is an effective approach to detecting aggressive behavior by detecting actions to modify the corresponding critical files [4]. If it is performed frequently, it will lead to a significant decrease in system performance; if rarely, then the effectiveness of such control may be low. Therefore, one of the main tasks in the implementation of mechanisms for controlling the integrity of file objects is the choice of principles and mechanisms for starting the integrity check procedure.

Another problem of integrity monitoring is the integrity control of the controlling program itself if the integrity control is implemented in software. All of this requires a certain additional study and the adoption of appropriate decisions depending, as a rule, on the features of specific information systems (ISs). Therefore, depending on the importance of the considered aspect of integrity and the data use scope, there are various methods and means to guarantee the integrity of the data under various possible threats. Thus, the correctness, non-distortion, and non-alteration of data can be ensured by methods and means of access control technologies based on formal models of integrity. Non-distortion of data during storage and transmission in information systems can be ensured through cryptographic primitives, such as digital signature, cryptographic hash functions, and message authentication codes. Parallel transaction technologies in multi-user systems also play an important role in ensuring the integrity of a database. The concept of a well-formed transaction is that users should not manipulate data arbitrarily, but only in ways that preserve the integrity of the database [5].

The objective of our paper was to present techniques and means that ensure the integrity of the main components of the database with a universal basis of relations (UBR) [6].

The expediency of researching precisely databases built on the basis of a schema with the universal basis of relations, implemented within the framework of the relational data model, is due to the fact that, first, this will make sure that the data and programs stored in them are secure from the point of view of their integrity. Second, based on their example (in view of the fact that databases with UBR can be used as an ordinary database, as a data warehouse for various subject domains (SDs), or as a configuration database of the dataspace management environment [7]), when applying certain new approaches, it becomes possible to develop a holistic solution that ensures the security of databases and data warehouses. Separate elements of such a solution can be used to protect databases and data warehouses with various models (relational, NoSQL, and NewSQL [8–14]) as well. All of this is important for the scientific community.

The main contribution of the authors is the development of techniques and means that ensure integrity of the main components of a database with the universal basis of relations in accordance with the recommendations of the Clark–Wilson model [15] as a methodological basis for building an integrity assurance system in information systems.

2. Related Works

Figure 1 shows a diagram of the main basic relations R^{sh} of the DB schema with UBR obtained by the authors of the article as a result of many years of research on the problem of creating a standard/universal data model, which has been discussed in the database community since the late 1980s [16–21]. Universal data models can provide effective solutions to many important data management problems [18]. The basic relations R^{sh} proposed by the authors have fundamental differences in the purpose, structure, and storage location of the description of the metadata of the simulated subject domain relative to the relations created in the traditional design technology of relational databases. Their number and structure do not depend on the data set (they are invariant to SDs), in contrast to the structure and number of basic relations of schemas developed using traditional technology. This makes it quite easy to adapt the database created in this way to changes in the SD. At the same time, the structure of DB schema relations remains unchanged. The pre-unlimited variety of SD elements is distributed over a fixed set of basic relations of the DB schema, while providing the possibility of the simultaneous storage and use of data from various significantly different SDs.

In order to more strictly and scientifically state the results of applied research related to ensuring the integrity of databases built on the basis of the schema with the universal basis of relations, it is advisable to use some security model, since it is known that security is easier to achieve if there is a clear model of what is to be protected and who is allowed to do what [22].

The use of formal security models makes it possible to formulate the requirements for creating secure systems (in this case, for the database) in a clearly defined form that corresponds to the security policy adopted in the organization. In general, a security model can be obtained from scratch using a mathematical model or by expanding an existing one. Although, neither of these approaches are easy, since they require the necessary formalization and re-proof [23]. Therefore, having analyzed, taking into account the peculiarities of the aspects under consideration, the well-known integrity models Biba [24], Clark–Wilson [15], and their application [1,2,23,25–30], as well as less well-known Goguen-Meseguer [31], Sutherland security [32], the Clark–Wilson model was taken as the basis. The Clark–Wilson model takes a multifaceted approach to ensuring integrity. This model does not require the use of a lattice structure, and instead of defining a formal state machine, it defines each data element and allows modifications only with a small set of programs [1]. The Clark–Wilson model is less of a specific security policy model, but rather a framework and guideline for formalizing security policies [29]. Data integrity, in accordance with the Clark–Wilson model, is achieved through [33] authentication, audit, well-formed transactions, and separation of duties.

Briefly characterizing the Clark–Wilson integrity model, the following can be noted. This model is based on triplets: "*Subject transaction not violating integrity object.*" Subjects, in accordance with this model, do not have direct access to objects. Objects can only be accessed through the *transformation procedure* (TP). TPs are the only procedures that are allowed to modify a *constrained data item* whose integrity is controlled by an IVP verification procedure (*integrity verification procedure*). IVP is a procedure that scans data items and confirms their integrity. Data whose integrity is not controlled by the security model is denoted as *unconstrained data items* (UDIs).

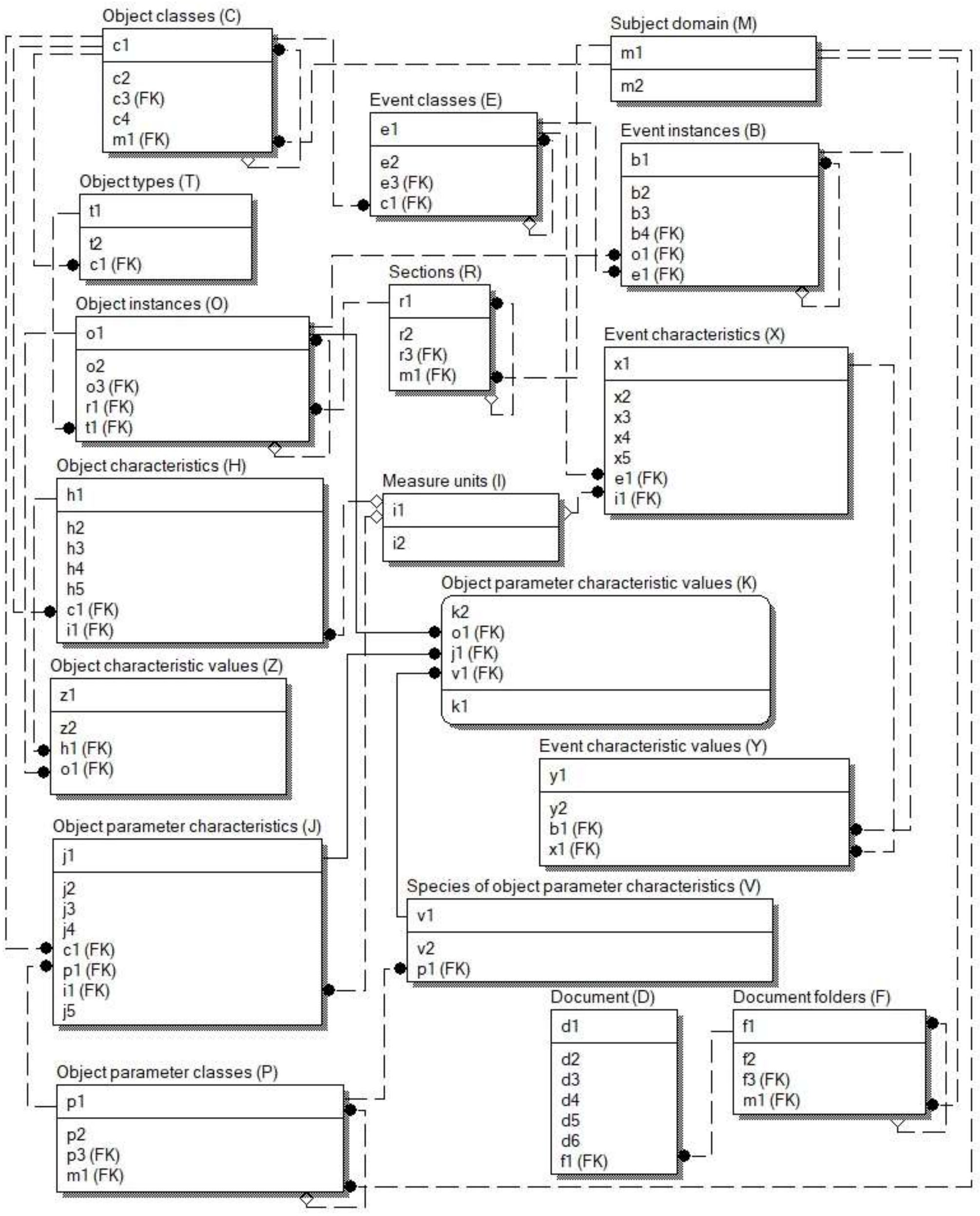

Figure 1. Diagram of the main basic relations R^{sh} of the DB schema with UBR.

The model consists of two sets of rules: Certification rules (C1–C5) and enforcement rules (E1–E4). Enforcement rules correspond to application-independent security functions, while certification rules allow application-specific integrity definitions to be included into the model. In other words, enforcement rules define the security requirements that must be supported by the protection mechanisms in the underlying system (in our case, it is a database management system (DBMS)). Certification rules define the security requirements that the application system should uphold (in this case, these are the proposed solutions within the framework of the DB with UBR schema, taking into account the features and capabilities of the DBMS on the platform on which it is implemented). Figure 2 shows a scheme of the application of these rules to data management.

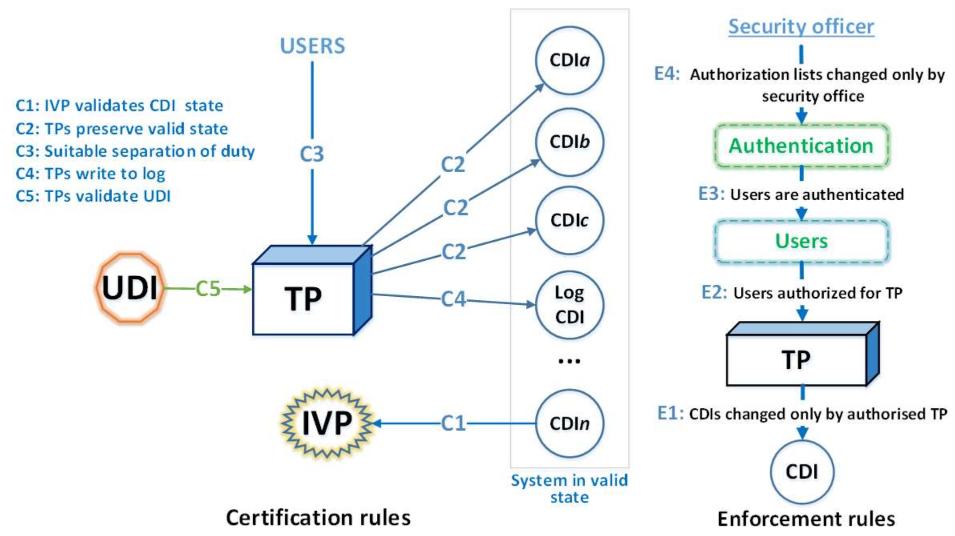
USERS
C1: IVP validates CDI state
C2: TPs preserve valid state
C3: Suitable separation of duty
C4: TPs write to log
C5: TPs validate UDI
C3
C2
C2
C2
C4
C5
C1
UDI
TP
IVP
CDIa
CDIb
CDIc
Log CDI
...
CDIn
System in valid state
Certification rules
Security officer
E4: Authorization lists changed only by security office
Authentication
E3: Users are authenticated
Users
E2: Users authorized for TP
TP
E1: CDIs changed only by authorised TP
CDI
Enforcement rules

Figure 2. Scheme of applying the rules of the Clark–Wilson model.

3. Applying the Clark–Wilson Model Recommendations to Ensure the Integrity of Databases with the Universal Basis of Relations

It is known that access to the data of any modern database is possible only through the DBMS. A traditional DBMS provides authentication, authorization, transaction, data management, logging, etc. Thus, to check whether the subject (user and process) has the necessary authorization to carry out the required operation in traditional DBMS, in the so-called database manager [9], there is a special module for authorization control. Therefore, the implementation of a DB with UBR on the platform of some selected relational DBMSs automatically leads to the fulfillment of the E3 rule requirement of the Clark–Wilson model, which instructs the system to authenticate all users (each subject) trying to perform any TP procedure.

According to the E4 rule, the access rights of subjects (taking into account their functional duties) to DB objects with UBR (processed CDI elements) can be assigned and changed only by specially authorized subjects (security officers, database administrators, and DB schema owners). For this purpose, the commands (statements) GRANT / REVOKE of the SQL standard are used. In addition, taking into account the peculiarities of the schema and the possibilities of using the DB with UBR [6,7], an additional mechanism for granting privileges was developed, implemented within the framework of RLS (*Row Level Security*) technology (also known as Fine Grained Access Control (FGAC) and Virtual Private Database (VPD)) [34–39], which required the introduction of some additional relations to the existing basic schema of the database with UBR:

– *User relation U*:

$$\begin{aligned} U = \; & \{(u_1, u_2, u_3) | u_1 \in U_1 \wedge u_2 \in U_2 \wedge u_3 \in U_3 \wedge \\ & ((\forall u_1 \forall u_2 \forall u_3 (\forall u'_2 \in U_2)(U_{pr}(u_1, u_2, u_3) \wedge U_{pr}(u_1, u'_2, u_3) \rightarrow u_2 = u'_2)\;) \wedge \\ & \quad (\forall u_1 \forall u_2 \forall u_3 (\forall u'_1 \in U_1)(U_{pr}(u_1, u_2, u_3) \wedge U_{pr}(u''_1, u_2, u_3) \rightarrow u_1 = u'_1)))\}, \end{aligned} \tag{1}$$

where U_1 is the set of user identifiers (subjects), U_2 is the set of user names, $U_{pr}(\ldots)$ refers to the predicates (predicate symbols) matching the relation U, and U_3 is the set of privileges granted to users for performing operations such as deletion, insert, update, select, as well as their combinations;

– *The relation of the access privilege distribution to the data of other users G*:

$$G = \{(g_1, g_2, g_3) | g_1 \in U_1 \wedge g_2 \in U_1 \wedge g_3 \in U_3\}. \tag{2}$$

The relation extension (2) is a set of tuples, each of which is associated with a specific data user/owner (g_1), which transmits its access privileges (g_3) to another authorized user (g_2).

As a rule, today, in relational DBMSs, individual records (fields and cells) are not specially protected, although there are examples known from practice when this is required. Therefore, in order to ensure such functionality, taking into account the invariance of the structure of the relations R^{sh} and based on the capabilities of the RLS technology, a special additional relation was also defined within the framework of the DB with the UBR schema. Namely, it is the relation of restrictions on access rights to a specific data element of the simulated SD:

$$A = \left\{ (a_1, a_2, a_3, a_4) \middle| a_1 \in U_1 \wedge a_2 \in U_2 \wedge a_3 \in R^{sh}_{name} \wedge a_4 \in R^{sh}_{ID} \right\},. \tag{3}$$

where R^{sh}_{name} is the set of names of database schema relations R^{sh} (Figure 1), and $R^{sh}_{ID} = \underset{i}{\cup} R^{sh}_i [K_{PK_i}]$ is the set of identifiers that are primary keys (K_{PK_i}) in the corresponding relations R^{sh}, access to which is limited for user $a_1 \in U_1$ with the name $a_2 \in U_2$.

In accordance with RLS technology, the following were defined:

– A set of declarative commands (RLS policies) that determine how and when to apply user access restrictions (in accordance with their functional duties, according to rule C3) to the tuples of the main relations R^{sh} of the DB schema with the UBR;
– A set of stored functions Ψ that are called when the conditions specified in the security policy (RLS policy) are performed;
– Predicates formed by Ψ functions that the DBMS automatically appends to the end of the WHERE clause of user-executed SQL statements.

Taken together, all of this can be represented as the implementation of the rules governing the access control to data of R^{sh} relations of the DB schema with UBR:

$$Sr = \left\{ R^{sh}_i, oper^j_i, policy^k_i, \Psi^l_i, attr^{\mu kl}_i, pat^{R^{sh}_i}_{contr} \right\}, \tag{4}$$

where $oper^j_i$ is j-th combination (from values select, update, delete, and insert) of allowed access operations (transformation procedures (TPs)) to the relation $R^{sh}_i \in R^{sh}$ (as one of the CDI elements); $policy^k_i$ is the name of the k-th RLS policy, which is applied to the base relation R^{sh}_i; $\Psi^l_i \in \Psi$ is the name of the l-th function that generates the predicate for the base relation R^{sh}_i; $attr^{\mu kl}_i$ is the value of the μ-th parameter for the k-th RLS policy and the l-th function; $pat^{R^{sh}_i}_{contr}$ is pattern of the commands for managing access to R^{sh}_i (an example of one of such patterns is given in [40] in the form of program code elements).

All of the above actions were taken so that the DBMS could control the admissibility of applying TP to the CDI elements and provide support for the list of TP transformation procedures required for specific users with an indication of the permissible set of processed elements CDI for each $TP_i \in TP$ and given subject ($s_j \in S$), in accordance with the requirements of rules E1 and E2 of the Clark–Wilson model.

For databases that support the relational data model, integrity constraints are ensured by ways of declarative and procedural support, each of which, in fact, leads to the creation and/or use of some program code that implements the constraint. The difference is only how the code is generated and where it is stored. At that, data integrity constraints must be preliminarily formally defined (declared) before the DBMS can ensure their implementation. In the case of operations that modify the contents of the database, in a traditional DBMS (in the DB manager), as a rule, there is a special data integrity checker module [9], which checks whether the requested operation satisfies all established data integrity constraints. Additionally, this module, taking UDI as input, activates TP, which either converts them to CDI or rejects (according to rule C5). The DBMS data integrity control module, controlling the admissibility of the application of transformation procedures TPs in relation to the list of elements $CDIs$ in accordance with rule E1, monitors the correctness of the implementation of all transformation procedures TPs (according to rule C2), in the sense that these procedures should not violate data integrity. Moreover, all of this takes into account the fact that the system must have procedures $IVPs$ capable of confirming the integrity of any CDI (rule C1).

When developing the main objects of the database schema with UBR, in order to protect the database from violation of the consistency of the data stored in it, the capabilities of both methods were used. Namely, in the created schema, using the integrity support means provided by the SQL language standard, implementations of the Pr^{sh} integrity constraints obtained as a result of the mapping were defined: $\gamma : \mathrm{Pr} \rightarrow \mathrm{Pr}^{sh}$ (where Pr is the set of integrity constraints that are specified in the data model with UBR (M_{ubr}) [6]).

The essence of declarative support for integrity constraints is the definition of constraints using the data definition language (DDL) of SQL. The means of declarative support for integrity were used to create the basic relations of the database schema with UBR to define such types of constraints as entity integrity, referential integrity, required (not null) data, and domain constraints. Namely, as known [8,9], the entity integrity is associated primarily with the uniqueness and irreducibility of the primary key. These integrity requirements were defined for all basic schema relations as a result of mapping (applying "*primary key*" and "*unique*" constructs of the corresponding SQL statements): $\gamma_{PK} : \mathrm{Pr}_{PK} \rightarrow \mathrm{Pr}^{sh}_{constr_{\text{primary\_key}}}$; $\gamma_{UK} : \mathrm{Pr}_{UK} \rightarrow \mathrm{Pr}^{sh}_{constr_{\text{unique}}}$ Below is an example of the result for such a mapping in the form of the main lines of DDL:

```
alter table MEAS_VALUES
add primary key (MEAS_TIME, MEAS_TYPE_ID, TYPE_ID, OBJECT_ID);.
```

As a result of the mapping: $\gamma_{FK} : \mathrm{Pr}_{FK} \rightarrow \mathrm{Pr}^{sh}_{constr_{\text{foreign\_key}}}$ (applying "*foreign key*" construction of the "*create/alter table*" operators), to ensure referential integrity, the foreign keys of the schema relations and the action strategies when deleting data were defined. As a result of the mapping: $\gamma_{not\_null} : \mathrm{Pr}_{not\_null} \rightarrow \mathrm{Pr}^{sh}_{constr_{\text{not\_null}}}$ (applying the "*not null*" specifier in the "*create/alter table*" statements), the constraints prohibiting the assignment of undefined values (*null*) to the corresponding attributes were set.

By mapping a set of integrity constraints of the data model with the universal basis of relations M_{ubr}, constraints for the feature attribute domains, data types of the characteristics of the objects, events, parameters of objects, and some others were defined in the database schema invariant to subject domains (as a result of mapping $\gamma_{dom} : \mathrm{Pr}_{dom} \rightarrow \mathrm{Pr}^{sh}_{constr_{\text{check}}}$, applying the "check" construction of the "*create/alter table*" operator). An example of the results for such mapping is as follows:

```
alter table EVENTS add check ((event_end_time is null) or ((event_end_time
is not null) and (event_end_time >= event_time)));.
```

However, not all integrity constraints could be implemented (thereby contributing to enforcing the requirements of rules C1 and C2) using declarative support. Therefore, along with the means of this way of implementing integrity constraints, procedural support means have found widespread use, such as triggers, stored procedures, and functions (for simplicity, sometimes united by the common name SQL procedures [41]), the mechanisms of which have been significantly expanded in many commercial DBMS in recent years [14,41]. Using procedural support means, the following integrity constraints ($\mathrm{Pr}^{sh}_{constr_{\text{proc}}}$) were implemented in the DB schema with UBR:

The constrains on possibility: Changing SD metadata entered into the corresponding relations of the schema (e.g., the maximum values of max_vals $\in at(R^{sh}_{\text{event\_prop\_types}})$) and the removal of the list values for the corresponding characteristics from the relations $R^{sh}_{pr\_vals}$, $R^{sh}_{ev\_pr\_vals}$, and $R^{sh}_{meas\_vals}$ if they are present in the relations associated with the data of the modeled SD [6];

- The constraints on the ability to enter new data that contradict the entered SD metadata (for relations R^{sh} associated with the SD data);
- Implementation of referential integrity for the schema relations R^{sh} associated with the relation R^{sh}_{docs} (a specific document from relation R^{sh}_{docs} is associated with a specific instance of the corresponding relation R^{sh} (Figure 1));

- The constraint of the maximum number of instances of objects (R^{sh}_{objects} relations) for a certain class of objects ($R^{sh}_{\text{obj\_classes}}$);
- The constraint of the maximum number of values ($R^{sh}_{\text{ev\_prop\_values}}$) that can be assigned to a certain event characteristic ($R^{sh}_{\text{event\_prop\_types}}$) for the event instance (R^{sh}_{events}) of the specific class;
- The constraints on the number of events (R^{sh}_{events}) that occur with one object instance (R^{sh}_{objects}):
  (a) At the same moment in time with one object instance, more than one event of the same class cannot occur;
  (b) One event that occurs with one object instance can have several subordinate events with different instances of objects occurring at the same time, but the specific event instance that occurs with the object instance of the certain class can have only one "event-owner";
- Generation of unique primary key values for schema relations R^{sh} and some others.

Figure 3 shows the scheme of applying techniques of declarative and procedural support for integrity constraints, which are used in the development of objects of the database schema with UBR to ensure the integrity of its data.

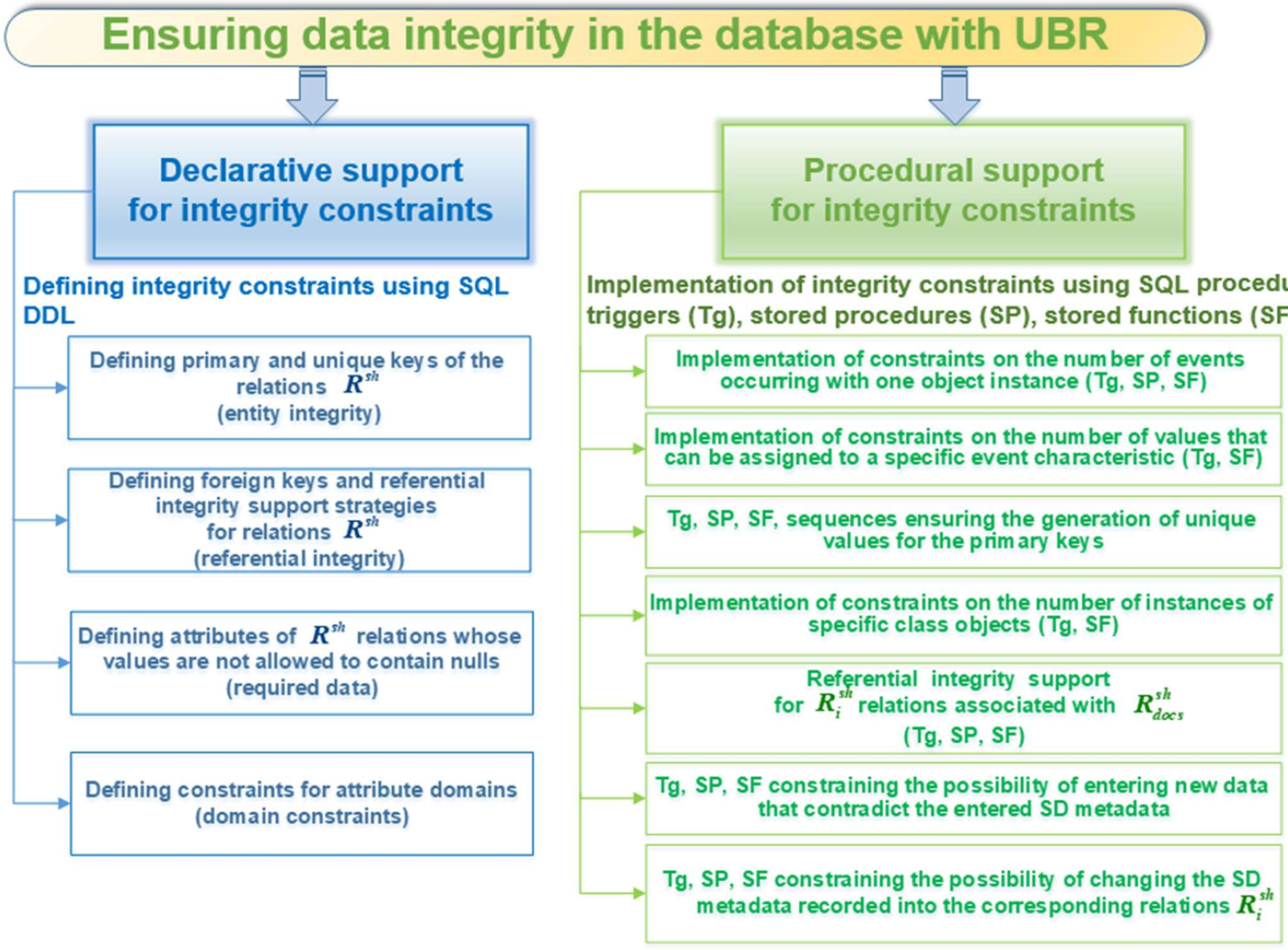
Ensuring data integrity in the database with UBR
Declarative support for integrity constraints
Defining integrity constraints using SQL DDL
Defining primary and unique keys of the relations R^{sh} (entity integrity)
Defining foreign keys and referential integrity support strategies for relations R^{sh} (referential integrity)
Defining attributes of R^{sh} relations whose values are not allowed to contain nulls (required data)
Defining constraints for attribute domains (domain constraints)
Procedural support for integrity constraints
Implementation of integrity constraints using SQL procedure triggers (Tg), stored procedures (SP), stored functions (SF)
Implementation of constraints on the number of events occurring with one object instance (Tg, SP, SF)
Implementation of constraints on the number of values that can be assigned to a specific event characteristic (Tg, SF)
Tg, SP, SF, sequences ensuring the generation of unique values for the primary keys
Implementation of constraints on the number of instances of specific class objects (Tg, SF)
Referential integrity support for R_i^{sh} relations associated with R^{sh}_{docs} (Tg, SP, SF)
Tg, SP, SF constraining the possibility of entering new data that contradict the entered SD metadata
Tg, SP, SF constraining the possibility of changing the SD metadata recorded into the corresponding relations R_i^{sh}

Figure 3. Scheme of using techniques of declarative and procedural support for integrity constraints to ensure data integrity in the DB with UBR.

In addition, taking into account the dual nature of database systems as an information product with two components (assets)—the actual data stored in the database, available for use, and DBMS software—as well as the possibilities of malicious impact on these assets, it is advisable to ensure the security of both of them. Therefore, below, we consider some aspects of ensuring the integrity of such important database objects performing data management as persistent stored modules (PSMs). These are specially designed programs, including SQL statements that are stored in a database, that can be invoked by applications and run within the DBMS. These include the aforementioned stored procedures, functions that can be combined into packages, triggers as a special kind of procedural code (a stored procedure that is called in response to the modification of the database contents [41]), and some others. Constant monitoring of these database objects (as CDI elements) is very important, since some of the attacks on the database (although not only on it, as, for example, you can attack the operating system through the vulnerabilities of the database server) can be detected precisely based on the modification analysis (intentional or accidental) of these objects (violation of their integrity) or their set (increase or decrease in their number) on the database server. Therefore, to ensure the possibility of monitoring the integrity of such stored modules, including those related to the DB schema with UBR, using the potential of the modern blockchain model, as shown in [42], the following have been developed:

– Structure;
– Techniques of forming the genesis and subsequent blocks;
– Verification methods (in the terminology of the Clark–Wilson model, this is IVP) of the PSM integrity, as well as two relations located in one of the privileged user database schemas, which are a mapping of the structure of blocks in the blockchain chain.

1 Relation of blockchain block headers R_{bch}:

$$R_{bch}(i_{id}, t, d_{DB}, n_{DB}, \quad n_{sh}, h_{root}, h_{block}, h_{p\_block}, n_{so}, w \Big| i_{id} \in \mathbb{N}^* \wedge t \in T \wedge d_{DB} \in Nm^{d_{DB}} \wedge \\ \wedge n_{DB} \in Nm^{DB} \wedge n_{sh} \in Nm^{sh_{DB}} \wedge h_{root} \in H_{Mr} \wedge h_{block} \in H_b \wedge \\ \wedge h_{p\_block} \in (H_b \cup \varnothing) \wedge n_{so} \in \mathbb{N}^* \wedge w \in W), \tag{5}$$

where i_{id} is the number of the i-th blockchain block; t is timestamp of block creation (T UTC Coordinated Universal Time); $Nm^{d_{DB}}$ is a set of database domain names; d_{DB} is the domain name of a specific database; Nm^{DB} is a set of database names; n_{DB} is the name of a specific database; $Nm^{sh_{DB}}$ is a set of names of the database schemas; n_{sh} is the name of a specific database schema (or *"genesis block"*); H_{Mr} is a set of hashes of Merkle roots; ($H_{Mr} = \{0,1\}^n$ is a set of all words of length n in the alphabet {0,1}); h_{root} is the hash of Merkle tree root of the i-th block ($i = 1 \ldots N_{bc}$, where N_{bc} is the total number of blockchain blocks); h_{block} is the hash of the header of the current i-th block; h_{p_block} is the hash of the header of the previous $(i-1)$-th block; H_b is a set of block hashes; ($H_b = \{0,1\}^n$); n_{so} is the number of controlled stored DB modules (as data items $CDIs$); $\mathbb{N}^*$ is a set of natural numbers without zero; W is a set of digital signatures ($w \in W$, $W = \{0,1\}^l$).

An example of a partially filled database table, which is a mapping of the relation R_{bch}, is given below (Table 1).

2 Relation of stored database modules (objects) R_{sp}:

$$R_{so}(i_{id}, p_k, \alpha_k, h_k | i_{id} \in \mathbb{N}^* \wedge p_k \in type_{so} \wedge \alpha_k \in Nm_{so} \wedge h_k \in H_{so}), \tag{6}$$

where Nm_{so} is a set of names of stored modules (objects), and H_{so} is a set of hashes of stored modules ($H_{so} = \{0,1\}^n$).

An example of a partially filled database table, which is a mapping of the relation R_{so}, is given below (Table 2).

Table 1. An example of a partially filled table of blockchain block headers. *

i_{id}	t	d_{DB}	n_{DB}	n_{sh}	h_{root}	h_{block}	h_{p_block}	n_{so}	w
296987922	21-APR-20 06.00.13.000000 PM +03:00	ua.xxx.com	WORKGR\ DESKTOP- QRRDTTA	genesis block	D420161F3 5294B0A64 7DD3E625 3C57AE25 8EC417D10 14EFC483A 66E7B6A9 1CE1	D420161F3 5294B0A64 7DD3E625 3C57AE25 8EC417D10 14EFC483A 66E7B6A9 1CE1		1	…
296987923	22-APR-20 02.34.01.575000 PM +03:00	ua.xxx.com	orcl	SYS	4DC69C66 60AF511F0 8D3F89FE89 9D19396269 676F657883 2EBC452EA 45F4AD56	442F64B40C 2CBA0E478 6DEC2FB9F A64C310C8 555F8E6F15 82E1651AE B7501CEB	D420161F3 5294B0A647 DD3E6253C 57AE258EC 417D1014EF C483A66E7 B6A91CE1	9799	…
296987924	22-APR-20 02.36.24.606000 PM +03:00	ua.xxx.com	orcl	user_1	3538FDE465 91936C2FF5 3D06909323 1E9F72C316 451629D44F AAE4AB221 FE2D1	F5415080C6 8CE7E671F5 262A968CE0 13B70C6B3B EC200C9E90 192D5AA22 ED6EC	442F64B40C 2CBA0E478 6DEC2FB9F A64C310C8 555F8E6F15 82E1651AEB 7501CEB	326	…
…	…	…	…	…	…	…	F5415080C6 8CE7E671F5 262A968CE0 13B70C6B3B EC200C9E90 192D5AA22 ED6EC	…	…

* The background color is used for better understanding.

Table 2. An example of a partially filled table of stored modules.

i_{id}	p_k	α_k	h_k
296987923	FUNCTION	AQ$_GET_SUBSCRIBERS	05A85236D79D0FFB86DEB 11B1F5D155C49B831A008 C6E96F4A389C3896540107
…	…	…	…

Access to these tables is limited: Only read/write and only to the owners of the corresponding schemas. In order to protect against unauthorized actions of a privileged user, as well as against illegitimate actions of attackers who illegally obtain the privileges of the owner of some schema with respect to the corresponding objects (modules), the proposed solution prescribes the creator of a specific database schema to sign "own" relevant data (see Table 1) with one of the modern digital signature algorithms. The result of the concatenation of hashed values (Merkle root hash, the timestamp, and the number of objects) is such signed data. The use of a hash tree structure, such as Merkle root, a digital signature mechanism to control the integrity and authenticity of objects stored in a specific database schema, is due to the objective need for rational use of resources, leading to savings for stored data and the computing resources of the processor.

As you know, the main disadvantage, usually mentioned for the Clark–Wilson model, is that IVP and related techniques are not easy to implement in real computer systems, in particular due to the fact that control of large amounts of information may be required, which is associated with a significant duration of the procedure IVP [30]. Thus, for example, in order to control the integrity of a specific stored module (as one of the CDI elements) in a specific database schema in the usual way, it is necessary to perform hashing and digital signature procedures, storing the corresponding data for each of them. The use of the hash tree structure allows ensuring the integrity control not only of the specific PSM being checked, but also of all other stored programs of the selected database schema, including the procedure that ensures the correctness of the formation of the values of Tables 1 and 2. Since this one data fragment is included in the general structure, changing at least one bit in it will entail a complete change in the value of the Merkle root. Therefore, Merkle trees are widely used for secure and efficient validation (control integrity) of large data structures [43–47].

On the DBMS server, the integrity control of the persistent stored modules, as described above, can be established with a certain periodicity as part of the audit with the recording of relevant information in the audit log with its subsequent analysis and taking effective measures. At that, the integrity check of a certain PSM can be initiated by any of the legitimate users of the system, who will contact the server with a corresponding request, which is described in more detail in [42].

An approach to the usage of the potential of the modern blockchain model can also be applied to control the data integrity of the relation R^{sh}_{docs}, in which various documents of the simulated subject domain can be stored. If necessary, it is also possible to provide control of the integrity of Tables 1 and 2. At that, some data of tables of Tables 1 and 2 can be converted into JSON format, after which a certain file will be formed from this data some file-ledger, which is distributed to all legitimate users. First, for the possibility of performing duplicate monitoring of unauthorized changes in stored database objects, and second, for the possibility for legitimate users of so-called lightweight nodes [43] to formulate correct queries to obtain information about the integrity of stored objects used in their applications. Using the concept of hash trees, and having certain data from the file-ledger, a legitimate user retains the ability to determine the fact of the presence of the object of interest stored in the database, as well as its integrity, by obtaining a small amount of data (as an authentication path in the Merkle tree) from the database server without the need to store or transfer a huge amount of blockchain data.

It is no secret that the audit procedure is equally important for creating a complete database security system. According to rule C4 of the Clark–Wilson model, each application of TP must be logged in a special item CDI, which is a log containing sufficient information to reconstruct a complete representation of each application of this transformation procedure, and available only for adding information to it. Therefore, to monitor the status, changes made to the database, user actions, in addition to using standard audit means of DBMS, on the platform of which the database schema with UBR is implemented, the developed special diagnostic functions implemented in the interpreter of the data model language (LDM) [48] are used. These functions can detect the introduction of incorrect data. For this purpose, triggers are also used that support the logging of operations performed in the database. In addition, for accountability of user actions, data from the log table of the modified data can be used [40]. Thanks to the information stored in the log table, which is automatically formed when the corresponding parameter of the stored procedure of the data model language interpreter is specified, the process of recovering incorrectly modified or lost data is simplified, and the procedure for determining the users, times, and nature of the modifications made by them is facilitated.

Thus, analyzing from the perspective of the Clark–Wilson model the possibilities of the above developed and implemented, including within the framework of the DB schema with UBR, techniques and means that ensure the integrity of the corresponding database elements of the CDI, we can conclude that they fully correspond to the main idea of the model. The basic theoretical principles of the integrity control policy lay out what needs to be done, and the mechanisms implemented define how these principles are achieved. Therefore, databases implemented based on a schema with UBR can be considered appropriate to the needs of databases protected from the point of view of integrity.

4. Conclusions

Using the recommendations of the Clark–Wilson model as a methodological basis for building an integrity assurance system in information systems, the authors developed techniques and means that ensure the integrity of the main components of a database with the universal basis of relations.

The proposed mechanisms are based on the provisions of the theory of relational databases, the RLS technology, the potential of the modern blockchain model, the capabilities of the SQL and LDM languages, as well as the DBMS on the platform on which DBs with UBR are implemented.

The implemented techniques and means, controlling changes of the stored CDI elements of the database with UBR, prevent their unauthorized change by authorized subjects and prevent changes by unauthorized subjects. As a result, the stored data and programs remain correct, unaltered, undistorted, and preserved. Consequently, databases built based on the UBR schema and supported by such mechanisms are protected in terms of integrity.

Author Contributions: Conceptualization, V.Y.; methodology, V.Y. and V.V.; software, V.Y., V.V., and K.W.; formal analysis, M.K.; investigation, V.Y., M.Y., V.V., and K.W.; writing—original draft preparation, V.Y., M.Y., and K.W.; writing—review and editing, V.Y., M.K. and M.Y.; funding acquisition, M.K. All authors read and agreed to the published version of the manuscript.

Funding: The research work reported in this paper was, in part, supported by the University of Bielsko-Biala, Poland, under program no. K18/1b/UPBJ/2019-2020.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chapple, M.; Stewart, J.M.; Gibson, D. *CISSP Certified Information Systems Security Professional Official Study Guide*, 8th ed.; Sybex, John Wiley & Sons, Inc.: Indianapolis, IN, USA, 2018.
2. Jueneman, R.R. Integrity controls for military and commercial applications. In Proceedings of the Fourth Aerospace Computer Security Applications, IEEE, Orlando, FL, USA, 12–16 September 1988; pp. 298–322. [CrossRef]
3. Shcheglov, A.I. *Protection of Computer Data from Unauthorized Access*; Nauka i Technika: St. Petersburg, Russia, 2004.
4. Jin, H.; Xiang, G.; Zou, D.; Zhao, F.; Li, M.; Yu, C. A guest-transparent file integrity monitoring method in virtualization environment. *Comput. Math. Appl.* **2010**, *60*, 256–266. [CrossRef]
5. Sandhu, R.S.; Jajodia, S. Data and database security and controls. In *Handbook of Information Security Management*; Auerbach Publishers: Boca Raton, FL, USA, 1993; pp. 481–499.
6. Yesin, V.I.; Karpinski, M.; Yesina, M.V.; Vilihura, V.V. Formalized representation for the data model with the universal basis of relations. *Int. J. Comput.* **2019**, *18*, 453–460. [CrossRef]
7. Yesin, V.I.; Karpinski, M.; Yesina, M.V.; Vilihura, V.V.; Veselska, O.; Wieclaw, L. Approach to Managing Data From Diverse Sources. In Proceedings of the 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Metz, France, 18–21 September 2019; pp. 1–6. [CrossRef]
8. Date, C.J. *An Introduction to Database Systems*, 8th ed.; Pearson Education Inc.: New York, NY, USA, 2004.
9. Connolly, T.M.; Begg, C.E. *Database Systems: A Practical Approach to Design, Implementation, and Management*; Pearson Education Limited: London, UK, 2015.
10. Sadalage, P.J.; Fowler, M. *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*; Pearson Education: London, UK, 2013.
11. Meier, A.; Kaufmann, M. *SQL & NoSQL Databases. Databases Models, Languages, Consistency Options and Architectures for Big Data Management*; Springer Fachmedien: Wiesbaden, Germany, 2019. [CrossRef]
12. Harrison, G. *Next Generation Databases: NoSQL, NewSQL, and Big Data*; Apress: Berkeley, CA, USA, 2015.
13. Pavlo, A.; Aslett, M. What's really new with NewSQL? *ACM SIGMOD Record* **2016**, *45*, 45–55. [CrossRef]
14. Garcia-Molina, H.; Ullman, J.D.; Widom, J. *Database Systems. The Complete Book*, 2nd ed.; Pearson Prentice Hall: Upper Saddle River, NJ, USA, 2009.
15. Clark, D.D.; Wilson, D.R. A Comparison of Commercial and Military Computer Security Policies. In Proceedings of the IEEE Symposium on Research in Security and Privacy (SP'87), Oakland, CA, USA, 27–29 April 1987; IEEE Press: Oakland, CA, USA, 1987; pp. 184–193.

16. Bernstein, P.A.; Dayal, U.; DeWitt, D.J.; Gawlick, D.; Gray, J.; Jarke, M.; Lindsay, B.G.; Lockemann, P.C.; Maier, D.; Neuhold, E.J.; et al. Future Directions in DBMS Research—The Laguna Beach Participants. *ACM SIGMOD* **1989**, *18*, 17–26. [CrossRef]
17. Bernstein, P.; Brodie, M.; Ceri, S.; DeWitt, D.; Franklin, M.; Garcia-Molina, H.; Gray, J.; Held, J.; Hellerstein, J.; Jagadish, H.V.; et al. The Asilomar report on database research. *ACM SIGMOD* **1998**, *27*, 74–80. [CrossRef]
18. Silverstone, L. *The Data Model Resource Book, Vol. 1: A Library of Universal Data Models for All Enterprises*; John Wiley & Sons, Inc.: Indianapolis, IN, USA, 2001.
19. Silverstone, L. *The Data Model Resource Book, Vol. 3: Universal Patterns for Data Modeling*; John Wiley & Sons, Inc.: Indianapolis, IN, USA, 2009.
20. Vyazilov, E.; Fedortsov, A.; Kobelev, A. Unification of data structure for field research, exploration and resources using of World Ocean. In Proceedings of the 10th All-Russian Scientific Conference "Digital Libraries: Advanced Methods and Technologies, Digital Collections", Dubna, Russia, 7–11 October 2008.
21. Vyazilov, E.D.; Fedortsov, A.A. Universal data storage model taking into account the life cycle of objects. In Proceedings of the Sixth All-Russian Open Annual Conference "Modern Problems of Remote Sensing of the Earth from Space", Moscow, Russia, 10–14 November 2008.
22. Tanenbaum, A.S.; Bos, H. *Modern Operating Systems*, 4th ed.; Pearson Education, Inc.: Upper Saddle River, NJ, USA, 2015.
23. Schott, M.; Krätzer, C.; Dittmann, J.; Vielhauer, C. Extending the Clark-Wilson security model for digital long-term preservation use-cases. In Proceedings of the SPIE 7542, Multimedia on Mobile Devices, San Jose, CA, USA, 27 January 2010. 75420M. [CrossRef]
24. Biba, K.J. *Integrity Considerations for Secure Computer Systems*; Mitre Corp: Bedford, MA, USA, 1977.
25. Whitman, M.E.; Mattord, H.J. *Principles of Information Security*, 6th ed.; Cengage Learning: Boston, MA, USA, 2017.
26. Katzke, S.; Ruthberg, Z. Report of the Invitational Workshop on Integrity Policy in Computer Information Systems (WIPCIS). NIST Special Publication 500-160. Available online: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication500-160.pdf (accessed on 24 August 2021).
27. Shockley, N.R. Implementing the Clark-Wilson integrity policy using current technology. In Proceedings of the 11th National Computer Security Conference, Baltimore, MD, USA, 17–20 October 1988; pp. 29–37.
28. Toapanta, S.M.T.; Trejo, J.A.O.; Gallegos, L.E.M. Analysis of Model Clark Wilson to Adopt to the Database of the Civil Registry of Ecuador. In Proceedings of the 21st conference of the Open Innovations Association FRUCT, Helsinki, Finland, 6–10 November 2017; pp. 513–518.
29. Gollmann, D. *Computer Security*, 3rd ed.; Wiley: Hoboken, NJ, USA, 2011.
30. Ge, X.; Polack, F.; Laleau, R. Secure databases: An analysis of Clark-Wilson model in a database environment. In *Advanced Information Systems Engineering. CAiSE 2004*; Persson, A., Stirna, J., Eds.; Lecture Notes in Computer Science, 3084; Springer: Berlin/Heidelberg, Germany, 2004; pp. 234–247. [CrossRef]
31. Goguen, J.A.; Meseguer, J. Security policies and security models. In Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, USA, 26–28 April 1982; pp. 11–20. [CrossRef]
32. Sutherland, D. A Model of Information. In Proceedings of the 9th National Computer Security Conference, Baltimore, MD, USA, 15–18 September 1986; pp. 175–183.
33. Van Tilborg, H.C.A.; Jajodia, S. *Encyclopedia of Cryptography and Security*, 2nd ed.; Springer Science & Business Media: New York, NY, USA, 2011. [CrossRef]
34. Row-Level Security in a Relational Database Management System/Curt Cotner, Gilroy, CA (US); Roger Lee Miller, San Jose, CA (US); International Business Machines Corporation, Armonk, NY (US)—N 10/233,397. US Patent 2004/0044655A1. 4 March 2004.
35. Row-Level Security in a Relational Database Management System/Curt Cotner, Gilroy, CA (US); Roger Lee Miller, San Jose, CA (US); International Business Machines Corporation, Armonk, NY (US)—N 12/242,241. US Patent 8,131,664 B2. 6 March 2012.
36. Row-Level Security in a Relational Database Management System/Curt Cotner, Gilroy, CA (US); Roger Lee Miller, San Jose, CA (US); International Business Machines Corporation, Armonk, NY (US)—N 15/343,568. US Patent 8,478,713 B2. 16 January 2018.
37. Feuerstein, S.; Pribyl, B. *Oracle PL/SQL Programming*, 6th ed.; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2014.
38. Kyte, T. *Expert Oracle*; Apress: New York, NY, USA, 2005.
39. Nanda, A.; Feuerstein, S. *Oracle PL/SQL for DBAs*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2005.
40. Yesin, V.I.; Yesina, M.V.; Rassomakhin, S.G.; Karpinski, M. Ensuring Database Security with the Universal Basis of Relations. In *Proceedings of the Computer Information Systems and Industrial Management. CISIM 2018*; Lecture Notes in Computer Science, 11127; Saeed, K., Homenda, W., Eds.; Springer: Cham, Switzerland, 2018; Chapter 42; pp. 510–522.
41. Groff, J.; Weinberg, P.; Oppel, A. *SQL. The Complete Reference*, 3rd ed.; McGraw-Hill Inc.: New York, NY, USA, 2010.
42. Yesin, V.I.; Yesina, M.V.; Vilihura, V.V. Monitoring the integrity and authenticity of stored database objects. *Telecommun. Radio Eng.* **2020**, *79*, 1029–1054. [CrossRef]
43. Bashir, I. *Mastering Blockchain: Distributed Ledger Technology, Decentralization, and Smart Contracts Explained*, 2nd ed.; Packt Publishing: Birmingham, UK, 2018.
44. Antonopoulos, A.M. *Mastering Bitcoin: Programming the Open Blockchain*, 2nd ed.; O'Reilly Media: Sebastopol, CA, USA, 2017.
45. Chapweske, J. Tree Hash Exchange Format. Available online: https://web.archive.org/web/20090803220648/http://open-content.net/specs/draft-jchapweske-thex-02.html (accessed on 24 August 2021).

46. Wei, W.; Yu, T. Integrity Assurance for Outsourced Databases without DBMS Modification. In Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy, Vienna, Austria, 14–16 July 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 1–16.
47. Niaz, M.S.; Saake, G. Merkle Hash Tree based Techniques for Data Integrity of Outsourced Data. In Proceedings of the 27th GI-Workshop Grundlagen von Datenbanken, Gommern, Germany, 26–29 May 2015; pp. 66–71.
48. Yesin, V.I.; Yesina, M.V. Language for universal data model. *Inf. Process. Syst.* **2011**, *5*, 193–197.