







Ensuring Database Security with the Universal Basis of Relations

Vitalii I. Yesin¹ , Maryna V. Yesina¹ , Serhii G. Rassomakhin¹ ,
and Mikolaj Karpinski² 

¹ V. N. Karazin Kharkiv National University,
4 Svobody Square, Kharkiv 61022, Ukraine

{v.i.yesin,m.v.yesina,rassomakhin}@karazin.ua

² University of Bielsko-Biala, 2 Willowa Street, 43-309 Bielsko-Biala, Poland
mpkarpinski@gmail.com

Abstract. The subject matter of the article is methods and means of the databases (DBs) security ensuring, built on the basis of the database scheme that is invariant to subject domains (SDs). The goal is to develop a substantiated approach that implements the complex use of various mechanisms ensuring the databases security built on the database schema with the universal basis of relations. The task: based on the analysis of existing database protection mechanisms supported by various database management systems (DBMSs), and features of the destination, construction of the database schema with the universal basis of relations, to develop and present in a systematized form the means and methods ensuring the databases security built on this DB schema. The following results were obtained: solving the problem of protecting databases as the most important corporate resource, in the process of creating database schema invariant to subject domains, special means were developed (in the form of implemented schema objects such as triggers, procedures, packages, tables, functions) and rules of their use, ensuring: access control to schema objects; data protection and hiding of objects; data integrity support; recovery of incorrectly modified or lost data; monitoring of the state, changes introduced into the database; logging user actions.

Keywords: Database with the universal basis of relations · Database security
Data protection · Access control · Security policy

1 Introduction

The database (DB), as the most important corporate information resource, should be properly protected. And in the case of using a database, built on the basis of a scheme with the universal basis of relations, that allows to store simultaneously in the fixed structure of the relations of its scheme the data of various subject domains (SDs) necessary for the company, organization or institution according to its multidisciplinary activities, as a corporate repository, the problem of ensuring the database security, as the protection of its data from undesirable disclosure-use (violation of confidentiality), falsification (integrity violation), loss or decrease in availability measure, becomes even

more actual. Herewith, it must be noted that the finished universal technique of integrated solution of the task ensuring of databases security today does not exist. This is explained by the variety of activities of enterprises, the structure of information systems, networks, database management systems (DBMS), data flows and ways to organize access to them. Therefore, in each specific situation, including when using a database with a universal basis of relations, this most important task requires an original approach, usually based on existing developments and solutions in this area.

2 Means and Methods for Protecting Databases Built on the Basis of the Scheme with the Universal Basis of Relations

The essence of the problem of ensuring the information security of databases is to develop methods and means that ensure the confidentiality, integrity, availability of their data under conditions of impact on them of any intentional or unintentional threats. Any threat should be considered as a potential possibility of security system violation that, if successful, can have some negative impact.

Existing approaches to solving this problem involves examining the database survey to identify possible threats that can lead to destruction, theft, data fraud, loss of confidentiality and data integrity, loss of availability and finding effective ways, means of confronting them. Herewith, first of all, the possibilities of means and methods of database protection that are supported by the database management system are analyzed.

The means and methods of databases protecting in various DBMS, on the platform of which the database scheme with the universal basis of relations [1–3] can be implemented, differ from each other. However, in varying degrees, often enough among them there are such as [4]: authorization, access control; views; backup and recovery; integrity support; encryption; the use of fault-tolerant hardware.

Below, without affecting the general security issues of computer systems, using the theory and the research results described in [4–12], we propose an approach implemented the complex use of various mechanisms ensuring databases security built on the basis of the database scheme that is invariant to subject domains (SDs) [1], taking into account the peculiarities of its structure.

It is known that to the violation of confidentiality leads both an intentional action aimed at the implementation of unauthorized access to the data and a random error of software or unskilled user action, which led to the disclosure of confidential information. Therefore, in the first place, the user authorization mechanism was analyzed to identify certain vulnerabilities in it and their subsequent elimination.

It is no secret that once a user get access right to database system, he can automatically be granted various privileges associated with his identifier. In particular, these privileges can include permission to access certain base (tables), virtual (views) relations of DB schema, its procedures, functions and other objects, as well as various actions with them. Such a method is sufficiently developed and flexible. It allows the database administrator to configure the access rights of users in accordance with their job responsibilities (the principle of access reasonableness). However, with it help, access is restricted only to named objects of the database schema, and, as a rule, only to a complete set of data that is provided to the respective users. While there is a need to control access at a lower level (although some DBMSs provide certain privileges for attributes of basic and virtual relations [5]). Using discretionary access control, it is impossible, for example, in full to restrict a subject access (a registered user) to only a part of DB relation tuples. And, taking into account the peculiarities of the database scheme with the universal basis of relations, which is invariant to subject domains and which allows to store data from different SDs in the fixed structure of the relations simultaneously, this is unacceptable. Thereby, analyzing the different approaches [4–8, 13] of the solution of this problem, it was concluded that it is advisable to use in this case of the mechanism of fine-grained access control, also known as row level security (RLS) [14].

That it was possible to take advantage of this mechanism benefits, as well as to simplify the procedure for registering the actions of database users (conducting an audit), ensuring auditability (one of the most important requirements of computer security), certain adjustments were made to the DB invariant scheme presented in [1] (in her base relations R^{sh}).

First, in each base relation of such database scheme $R_i^{sh} \in R^{sh}$ the following attributes were added: user identifier ($u_i \in at(R_i^{sh})$); the tuple recording time ($t_i^{ins} \in at(R_i^{sh})$); tuple component update time ($t_i^{upd} \in at(R_i^{sh})$). Secondly, since the specific user (its identifier) $u^j \in U_1$ is associated with the value of the attribute $u_i(U_1$ – the set of user identifiers) for which access privileges to the tuples of the base relations should be defined $R^{sh} - p_j \in U_3$ (where 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 was defined whose extensions include data on the corresponding database user names, their identifiers and corresponding their access privileges. In a formalized form, such relation, referred to as the relation of users, can be represented as a subset of the Cartesian product $U_1 \times U_2 \times U_3$:

$$U = \{(u_1, u_2, u_3) | u_1 \in U_1 \wedge u_2 \in U_2 \wedge u_3 \in U_3\}, \quad (1)$$

where U_2 is the set of user names; $u_1 = u^j$; $u_3 = p_j$;

$$p_j = p_j^{gl} \cup p_j^{del} \cup p_j^{upd} \cup p_j^{sel} \cup p_j^{ins}, \quad (2)$$

$$\begin{aligned}
 p_j^{gl} &= \begin{cases} p^{gl} \in P_{user}, & \text{if } u^j \text{ has the privilege to access data } \forall u^k, k = 1 \dots |U_1|; \\ p^{none} = 0 (p^{none} \in P_{user}), & \text{else;} \end{cases} \\
 p_j^{del} &= \begin{cases} p^{del} \in P_{user}, & \text{if } u^j \text{ has the privilege to delete his data;} \\ p^{none} = 0 (p^{none} \in P_{user}), & \text{else;} \end{cases} \\
 p_j^{upd} &= \begin{cases} p^{upd} \in P_{user}, & \text{if } u^j \text{ has the privilege to update his data;} \\ p^{none} = 0 (p^{none} \in P_{user}), & \text{else;} \end{cases}
 \end{aligned}$$

similarly for the privilege p_j^{sel} that allows to j user (u^j) to select the data available to him from the base relations R^{sh} , and data insert privilege – p_j^{ins} ; $P_{user} = \{p^{none}, p^{del}, p^{upd}, p^{sel}, p^{ins}, p^{gl}\}$ is user privilege domain.

The result of mapping the relation U to the base relation $R_U^{sh} \in R^{sh}$ of the database scheme that is invariant to SDs is represented in the form of the main lines of data definition language code of the ISO SQL standard used in the CREATE (ALTER) TABLE operators:

```

USER_ID    NUMERIC(12) PRIMARY KEY, -- user id code(PK)
USER_NAME  VARCHAR(30) not null,    -- user name
USER_ISUD  NUMERIC(2)  not null     -- user privileges.

```

Thirdly, to implement the possibility that the grantor (by one authorized user u^j) transferred the privileges belonging to him to other authorized users (grantees), the new mechanism was developed, since the traditional way (using GRANT command of the SQL standard) was not fully suitable for the database scheme with the universal basis of relations, taking into account its destination and structure of relations R^{sh} . Such mechanism implemented within RLS technology required the development of a new relation for the database scheme that is invariant to SD, namely so-called of the access privilege distribution relation to the data of other users. In a formalized form, it can be represented as a subset of the Cartesian product $U_1 \times U_1 \times U_3$:

$$G = \{(g_1, g_2, g_3) | g_1 \in U_1 \wedge g_2 \in U_1 \wedge g_3 \in U_3\}. \tag{3}$$

The relation (3) extension is a set of tuples, each of which is associated with a specific data owner (g_1), which transmits its access privileges (g_3) to another authorized user (g_2).

The result of mapping this relation to the base schema relation ($R_G^{sh} \in R^{sh}$) is shown below in the form of the following main lines of SQL code used in CREATE (ALTER) TABLE operators:

```

GRANT_ID   NUMERIC(12) PRIMARY KEY, -- primary key (PK)
GRANTOR    NUMERIC(12) not null,    -- user-owner of data
GRANTEE    NUMERIC(12) not null,
USER_ISUD  NUMERIC(2)  not null     -- user privileges.

```

Further in accordance with the RLS technology were determined:

- a set of declarative commands (RLS policies) that determine how and when have to apply users access restrictions to the tuples of the schema base relations R^{sh} ;
- a set of stored functions Ψ (combined in a package) that are called when the conditions specified in the security policy (RLS policy) are performed;
- predicates formed by functions Ψ that the DBMS automatically appends to the end of the WHERE clause of user-executable SQL statements (the consumption of system resources depends on the correct predicate formation).

In the aggregate, all this can be represented as the implementation of rules for protecting relations R^{sh} and formalized in the form of the following expression:

$$Sr = \{R_i^{sh}, oper_i^j, policy_i^k, \Psi_i^l, attr_i^{\mu kl}, pat_{contr}^{R_i^{sh}}\}, \quad (4)$$

where $oper_i^j$ is the j -th combination (from the values: select, update, delete, insert) of the allowed access operations to the relation $R_i^{sh} \in R^{sh}$; $policy_i^k$ is name of the k -th RLS policy, which is applied for the base relation R_i^{sh} ; $\Psi_i^l \in \Psi$ is the name of the l -th function (specifying the package name) that generates the predicate for the base relation R_i^{sh} ; $attr_i^{\mu kl}$ is the value of the μ -th parameter for the k -th RLS policy and the l -th function; $pat_{contr}^{R_i^{sh}}$ is pattern of the commands for managing access to R_i^{sh} (for implementation of the security policy).

Herewith, it should be noted that, for example, when implementing the database scheme with the universal basis of relations on the Oracle DBMS platform, in order to enhance the capabilities of RLS technology, guided by the recommendations [7, 8], it is also expediently to use the mechanism of so-called application contexts (named set of pairs “parameter-value”). The idea underlying the use of contexts is simple enough, but, at the same time, it allows to provide serious protection. The list of variables in memory (context), whose values are bound to sessions, is determined. Herewith, the session can get the current values of these variables, calling a special function, and variables in the context can be set only by calling a procedure associated with this context.

An example of the pattern of access control commands (of security policy implementation for Oracle DBMS) is given below.

```

create or replace package rls_utils is
...
function get_predicate(dml_stmt_p number, object_name_p
varchar2) return varchar2;
function get_access_rule_predicate(schema_p in varchar2,
object_p in varchar2) return varchar2;
...
end;
/
create or replace package body rls_utils is
...
function get_access_rule_predicate
(schema_p varchar2, object_p varchar2) return varchar2
as
begin
return (get_predicate(DML_access_rule, object_p));
end;
...
end;
/
{
begin
dbms_ols.add_policy(object_schema => '{user_name}',
object_name => '{table_name}',
policy_name => '{RLS_user_name_access_rule}',
function_schema => '{user_name}',
policy_function =>
'{RLS_UTILS.GET_access_rule_PREDICATE}',
statement_types => '[SELECT] [,INSERT] [,DELETE]
[,UPDATE]',
update_check => {TRUE|FALSE},
enable => {TRUE|FALSE},
static_policy => {TRUE|FALSE}
);
end;
/
}

```

The following symbols are used in the above pattern:

- variables (in bold italic font): *user_name* is user name; *table_name* is base relation name $R_i^{sh} \in R^{sh}$ to be protected by policy; *access_rule* is combinations of access operations (select, update, delete, insert – $oper_i^j$, expression (4)) to the relation specified in *table_name*;

- parameters of the *add_policy* procedure of the standard Oracle package `dbms_rls`: *policy_name* is the name of the RLS policy that is applied to the base relation $R_i^{sh} \in R^{sh}$; *policy_function* is the name of the function owner that returns the condition; *policy_function* is the name of the function (with the name of the package) which generates a predicate for the base relation $R_i^{sh} \in R^{sh}$; *statement_types* is statement types to which the policy applies; *update_check* is optional argument for INSERT or UPDATE statement types (the default is false). Setting *update_check* to true causes the server to also check the policy against the value after INSERT or UPDATE; *enable* is a parameter indicating the activation of the policy immediately after its addition (the default is true); *static_policy* is a parameter (the default is true). If it is set to true, the server assumes that the policy function for the static policy produces the same predicate string for anyone accessing the object, except for SYS or the privileged user who has the EXEMPT ACCESS POLICY privilege.
- the braces are {}, the square brackets are [], the symbol | correspond to the notation taken from the extended Backus-Naur notation;
- all other alphanumeric characters are either language keywords, either by standard package names or by accepted string literals.

As a rule, today in relational DBMS separate records, from the point of view of the access organization to them of various users, are not specially protected, although there are examples known from practice when it is required [13, 15]. Therefore, in order to provide such functionality, based on the capabilities of the above fine-grained access control mechanism, taking into account the predetermined relation R^{sh} structure, a special additional relation of the DB scheme with the universal basis of relations was developed, the data of which is used by the function forming the predicate. This relation, referred to as the relation of access restrictions to a specific data item, can be presented in a formalized form as a subset of the Cartesian product $U_1 \times U_2 \times R_{name}^{sh} \times R_{ID}^{sh}$:

$$A = \{(a_1, a_2, a_3, a_4) | a_1 \in U_1 \wedge a_2 \in U_2 \wedge a_3 \in R_{name}^{sh} \wedge a_4 \in R_{ID}^{sh}\}, \quad (5)$$

where R_{name}^{sh} is the set of schema relations R^{sh} names; $R_{ID}^{sh} = \cup_i R_i^{sh}[K_{PK_i}]$ is a set of identifiers that are primary keys (K_{PK_i}) in the corresponding relations R^{sh} , access to which is limited to a user with an identifier $a_1 \in U_1$ and a name $a_2 \in U_2$.

The result of mapping this relation to the base relation ($R_A^{sh} \in R^{sh}$) of the database scheme that is invariant to SD is represented as the following main lines of SQL code:

```

USER_ID      INTEGER not null,  -- user ID
USER_NAME    VARCHAR(255) not null, -- user name
CLASS_ID     INTEGER not null, -- object ID in TABLE_NAME

TABLE_NAME   VARCHAR(255) not null -- table name ( $R_{name}^{sh}$ )
primary key (CLASS_ID, TABLE_NAME) -- PK.

```

The packet body fragment forming the context attributes and the predicate string is shown in [10].

Herewith, in general, it is necessary to understand that the implementation of the access mechanism of different users to specific individual data elements leads to an increase in the total time required to process the corresponding requests to the database.

In order to organize access to data and system resources in the development of the database schema with the universal basis of relations, its following objects were also identified: roles designed to simplify the management of system and object privileges; synonyms required to specify an alternate object name of the database; profiles as a named set of resource limits and password parameters that restrict database usage for a user.

To protect important information stored in the database, access to it from one side should be limited, and on the other side it is advisable to encrypt it. Data encryption is a key component in implementing the principle of multilevel protection. The desire to reduce the risk of data confidentiality loss, including due to insider threats of privileged users, has become a motivated start for: developing mechanisms that provide the possibility of effective use for the protection of cryptographic primitives supported by the DBMS (if there are any and they satisfy the consumer of the information product); developing their own cryptographic protection means (if they are not available or they do not fully meet the requirements of the consumer of the information product); or for their complex use.

For this purpose, a package of subprograms, a technique for its application was developed. Also recommendations on the use of existing technologies of information encryption and hiding have been determined.

The developed package of subroutines and the technique of its application provide for the integrated use of both the supported cryptographic primitives by the DBMS (for example, for Oracle DBMS it is cryptographic algorithms AES, 3DES168, RC4; cryptographic hash algorithms: MD5, SHA-1, SHA-2; keyed hash (MAC) algorithms: HMAC_MD5, HMAC_SH1, HMAC_SH512 and others) and the symmetric block cipher “Kalyna” from the national Ukrainian encryption standard DSTU 7624:2014.

Large amounts of data and discretionary access to information stored in a database based on a database schema with the universal basis of relations to a certain extent complicate the implementation of an effective mechanism when the content is pre-decrypted and then, after use, is encrypted back. Therefore, in order to preserve the habitual tools and operating procedure in the proposed approach, it is recommended to use the so-called “transparent encryption” method, in which the information is automatically decrypted when reading from the medium (if the correct key was entered) and automatically encrypted during recording. So transparent data encryption (TDE) allows you to selectively encrypt vulnerable data that is stored in database files, as well as all stream file components, such as redo logs, archive logs, backup tapes. The TDE technique is inherent in various DBMSs (IBM patent 7426745 [16]). The main purpose of TDE is to protect vulnerable data in the appropriate operating system files.

In addition, without resorting to the encryption procedure, for several reasons: observance of intellectual property rights; commercial value; the code provides the solution of problems of protection and distribution of data access rights; the inadmissibility of code modification by other users (especially after installing the software to the consumer of the information product, that the latter had no reason to declare after his own modification (in fact, hacking) about the inoperability and unreliability of software), code of the main procedures, functions, packages, triggers of the database

schema with the universal basis of relations is advisable to hide. For this purpose, it is suggested to use the corresponding DBMS tools. For example, in Oracle DBMS, such the most suitable means is the special utility WRAP. This utility allows to hide PL/SQL code of main objects of the database schema quite simply and effectively, transforming this code into an unreadable form, which in its turn is uniquely understood by the server. The server can compile and execute it. Herewith the code is changed, and not encrypted with complex algorithms. Such a substitution does not greatly affect the performance, unlike the encryption/decryption procedure. However, using this mechanism, it should be kept in mind that if you need to change the source code, you will have to change the original again, hide it with the utility and load the hidden version into the database.

In addition to this mechanism, special pipelined-functions were developed to hide the composition and structure of the base and virtual relations of the database schema with the universal basis of relations. These functions with the parameter in the form of a meta-description line of the data model language (LDM) [17, 18] can be used in SQL statements (in FROM clause) instead of base and virtual relations of the database schema with the universal basis of relations.

An example of using pipelined-function:

```
SELECT '<ClassO>=' || COLUMN_VALUE as name
FROM TABLE(get_spis_metadata('{<ClassO>=*.*; }'))
ORDER BY name;
```

where *get_spis_metadata* is a pipelined function with a parameter in the form of a meta-description line of the LDM that is used in the SELECT statement as a virtual table to obtain a list of certain requested data of the considered SD.

It is known that the loss of database data integrity can have the most serious consequences for the future work of the organization. Therefore, in the database scheme with a universal basis of relations, appropriate means of maintaining the data integrity were realized, which were considered in detail in [1], which contribute to the overall security of the database, preventing the possibility of data transition to an inconsistent state, thereby excluding the threat of receiving erroneous or incorrect results.

Using databases based on a database scheme with the universal basis of relations, it is recommended as one of the mechanisms that contribute to increasing the DB availability, to perform periodically backup their contents and organize the storage of created copies in places provided with the necessary protection. Today, almost any modern DBMS provides backup tools that allow you to restore a database. At the same time, in addition to the capabilities of standard backup and recovery tools, it became expedient to define one more relation of the database scheme with the universal basis of relations. Namely, a relation, referred to as a log of changed data. In a formalized form,

this relation can be represented as a subset of the Cartesian product $OS_{user} \times IP \times U_2 \times DB_{name} \times L_{DM} \times T_{DB} \times T_{DDB} \times Op \times P_{name}$:

$$L = \{(l_1, l_2, l_3, l_4, l_5, l_6, l_7, l_8, l_9) | l_1 \in OS_{user} \wedge l_2 \in IP \wedge l_3 \in U_2 \wedge l_4 \in DB_{name} \wedge l_5 \in L_{DM} \wedge l_6 \in T_{DB} \wedge l_7 \in T_{DDB} \wedge l_8 \in Op \wedge l_9 \in P_{name}\}, \quad (6)$$

where OS_{user} is the set of device names (host-machines of clients) from which the session was activated; IP is the set of IP addresses of the devices from which the session was activated; DB_{name} is set of database names; L_{DM} is the set of meta description lines of LDM, leading to a change in the data stored in the database; T_{DB} is the set of times when changes were made to the current database; T_{DDB} is the set of times of data output to other databases (when replication, distribution of data in a distributed system); Op is the set of statement that lead to the modification of data stored in the database ($Op = \{insert, delete, update\}$); P_{name} is the set of procedure names of the LDM interpreter [18] ($P_{name} = \{Proc_{metadata}, Proc_{data}\}$).

The result of mapping this relation to the base relation of the database schema with the universal basis of relations ($R_L^{sh} \in R^{sh}$) is presented below in the form of the main lines of SQL code:

```

HOST          VARCHAR2(255) not null, -- host name
IP_ADDRESS    VARCHAR(255) not null, -- IP address
SESSION_USER  VARCHAR(255) not null, -- user name
DB_NAME       VARCHAR(255) not null, -- database name
NAME_STR_METADATE VARCHAR(2000) not null, -- lines of LDM
TIME_WRITE_LOC DATE not null, -- time for current DB
TIME_INTO_GLOB  TIMESTAMP(9),
NAME_PROC      VARCHAR(100) not null, --
{Proc_{metadata}, Proc_{data}}
OPERATION     VARCHAR(10) not null, -- ins, del, update
primary key   (NAME_STR_METADATE, TIME_WRITE_LOC).
    
```

Thanks to the information stored in the log table, which is automatically formed when the corresponding parameter of the stored procedure of the LDM interpreter is specified [18], the process of recovering incorrectly changed or lost data is simplified, and the procedure for determining users, times and the nature of their changes is facilitated. In addition, the information from the log-table of the changed data can be used in distributed systems when data is propagated (replicated).

It is no secret that an audit procedure is no less important for creating a complete database security system. Actions with critical data should be logged. Therefore to monitor the status, changes introduced to the database, user actions, in addition to using standard DBMS audit tools, on the platform of which the database scheme with the universal basis of relations was implemented, special diagnostic functions, including dynamic analyzers of code coverage implemented in the LDM interpreter, capable of

detecting entering incorrect data, as well as triggers that support the logging of operations performed in the database have been developed. Also, for accountability of user actions, data from the log table of the changed data, as discussed above, can be used.

Thus, solving the problem of protecting corporate databases built on the basis of the database scheme with the universal basis of relations, from possible threats, special means (in the form of implemented scheme objects) and the rules for their use were developed in the process of creating this database scheme. These means and rules are based both on common methods and tools supported by the DBMS, on the platform of which the proposed scheme is implemented, and on its own mechanisms developed within the framework of creating this scheme. The means and methods implemented in the DB schema with the universal basis of relations to ensure the security of databases are shown in a systematized form in Fig. 1.

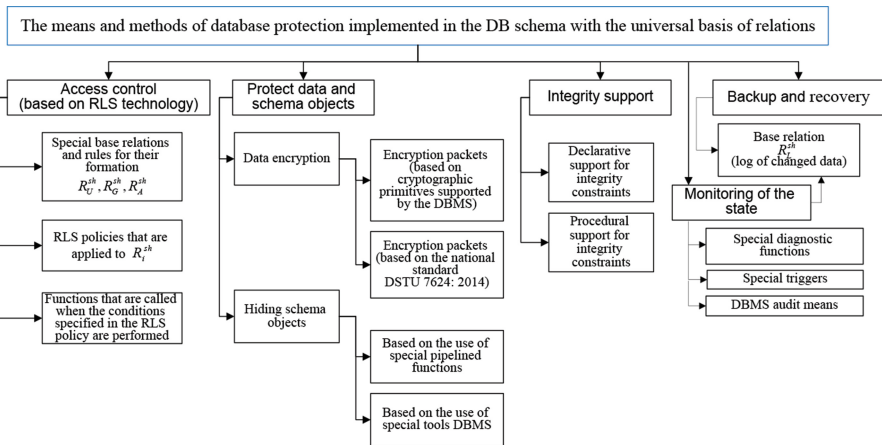


Fig. 1. The means and methods implemented in the DB schema with the universal basis of relations to ensure the security of databases

3 Conclusions

1. To ensure the security of data in databases built on the basis of the database scheme with the universal basis of relations, an approach is proposed of the integrated use of common methods and tools supported by the DBMS, on the platform of which this scheme is implemented, as well as its own mechanisms developed in the framework of creation of the DB schema that is invariant to SDs.
2. Solving the problem of protecting databases as the most important corporate resource, in the process of creating database schema invariant to subject domains, special means were developed (in the form of implemented schema objects such as triggers, procedures, packages, tables, functions) and rules of their use, ensuring: access control to schema objects; data protection and hiding of objects; data integrity support; recovery of incorrectly modified or lost data; monitoring of the state, changes introduced into the database; logging user actions.

3. Implemented in DB scheme with the universal basis of relations, means and methods of protection allow you to control access to data up to a specific element.
4. The practice of using databases built on the basis of the database scheme with the universal basis of relations, for information systems of different subject domains, in the projects of which it was required to organize reliable, safe storage, adaptation to changes occurring in SD and legislation, timely processing of data, showed that they have a sufficiently high degree of controllability of access to data, cryptographic protection of data, reliability, stability.

References

1. Esin, V.I.: Invariantnaya k predmetnym oblastyam shema bazy dannyh i ee otlichitelnye osobennosti. Radiotekhnika: nauch.-tehn. zhurnal. 193, 133–142 (2018). (in Russian)
2. Esin, V.I.: Model dannyh s universalnoj fiksirovannoj strukturoj. In: Materiali mizhnarodnoyi naukovoyi konferenciyi, TAAPSD 2014, pp. 112–116. FO-P Aleksandrova M.V., Kirovograd (2014). (in Russian)
3. Esin, V.I., Pergamencev, Y.A.: Tehnologiya proektirovaniya modeli predpriyatiya na osnove universalnoj modeli dannyh. <http://www.citforum.ru/database/articles/udm/>. Accessed 26 Mar 2018. (in Russian)
4. Connolly, T.M., Begg, C.E.: Database Systems: A Practical Approach to Design, Implementation, and Management, 6th edn. Pearson Education Limited, Harlow (2015)
5. Groff, D.R., Vajnberg, P.N., Opper, E.D.: SQL: polnoe rukovodstvo, 3-e izd., Per. s angl., OOO Izdatelskij dom “Vilyams”, Moskva (2015). (in Russian)
6. Date, C.J.: An Introduction to Database Systems, 8th edn. Addison-Wesley, Pearson (2004)
7. Kajt, T.: Oracle dlya professionalov: Per. s angl. OOO «DiaSoftYuP», Sankt-Peterburg (2003). (in Russian)
8. Nanda, A., Fejershtejn, S.: Oracle PL/SQL dlya administratorov baz dannyh. Per. s angl. Simvol-Plyus, Sankt-Peterburg (2008). (in Russian)
9. Grachev, V.M., Esin, V.I., Polukhina, N.G., Rassomakhin, S.G.: Data security mechanisms implemented in the database with universal model. Bull. Lebedev. Phys. Inst. **41**(5), 123–126 (2014). <https://doi.org/10.3103/s1068335614050029>
10. Esin, V.I., Esina, M.V.: Osobennosti zashity dannyh v bazah dannyh s universalnoj modelyu. Prikladnaya radioelektronika **10**(2), 226–232 (2011). (in Russian)
11. Soroka, L.S., Esin, V.I., Esina, M.V.: Tradicionnye metody i sredstva zashity dannyh, realizovannye v baze dannyh s universalnoj modelyu dannyh. Visnik Akademiyi mitnoyi sluzhbi Ukrainy. Ser.: Tehnichni nauki **2**(44), 7–12 (2010). (in Russian)
12. Esin, V.I., Yurasov, V.G.: Zashita dannyh v baze dannyh s universalnoj strukturoj. Informaciya i bezopasnost. Voronezh, Voronezhskij gosudarstvennyj tehniceskij universitet. **17**(2), 180–187 (2014). (in Russian)
13. Fedorov, A.V., Pyankov, V.M., Vihlyancev, P.S., Simonov, M.V.: Sistema razgranicheniya dostupa k dannyh na urovne zapisej i yacheek. Zashita informacii INSIDE **3**, 2–4 (2012). (in Russian)
14. Patent 8,131,664 B2, United States, 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 (2012)
15. Homonenko, A.D., Cyankov, V.M., Malcev, M.G.: Bazy dannyh. KORONA print, Sankt-Peterburg (2004). (in Russian)

16. Methods and systems for transparent data encryption and decryption, United States Patent 7426745. <http://www.freepatentsonline.com/7426745.html>. Accessed 26 Mar 2018
17. Esin, V.I., Esina, M.V.: Yazyk dlya universalnoj modeli dannyh. Sistemi obrobki informaciyi. **5**(95), 193–197 (2011). (in Russian)
18. Esin, V.I., Esina, M.V.: Interpretator yazyka dlya universalnoj modeli dannyh. Nauka i tehnika Povitryanij Sil Zbrojnih Sil Ukrajini **2**(6), 140–143 (2011). (in Russian)