Functional Testing Criteria Applied in a Database Project

Dianne Dias Silva, Edmundo Sérgio Spoto, Leandro Luís Galdino de Oliveira
Instituto de Informática (INF)
Universidade Federal de Goiás (UFG)
Goiânia, Brazil
Emails: {diannesilva, edmundo, leandroluis}@inf.ufg.br

***Abstract*—This paper reports the application of the functional testing criteria for a database project in an IT company, aiming to explore the main features that exist in the project of the company. The paper presents a set of required elements based on the functional testing and the results of test cases analyzed in the project in question. The article also presents the criteria that were most effective in detecting faults in the organization of the database project.**

Keywords-Software Testing; Database Testing; Functional Software Testing; Functional Testing Criteria in a Database Project.

I. INTRODUCTION

Gradually, software has come to play a relevant activity in everyday society, so its reliability can not be ignored. The reliability criteria are specified by parameters of quality and as a result, sets of Software Quality Assurance (SQA) activities can be defined by supportting the software development process.

In this context, the activity of software testing should be started in parallel with the software design, requiring good planning, since the determination of testing criteria to be used, the definition of Test Cases (TC) and mass test data, come from this stage of the development cycle.

The testing criteria determine the Required Elements (RE), which must be tested because in general, the exhaustive testing is not feasible. Also, the RE are associated with testing techniques that explore different aspects and functionality of the software, relating to the functional technique (Equivalence Class Partitioning and Boundary Value Analysis), the statement structure of the project code belonging to the structural technique (Based-on-Control-Flow and Based-on-Data-Flow) and typical faults inserted into the software during its implementation caused by error based techniques (Error Seeding and Mutation Analysis) [5].

In addition, the growing usage of database applications in both small and large organizations, requires that relevant characteristics of database project, as cardinality, domain attributes, functional dependency, among others, are treated.

However, the techniques, strategies and tools for testing application database are scarce. Chays et al. [6] presented a number of important features of database project testing to be explored both in the development and the in the operation stages of the project.

The testing criteria for database projects used in this paper were chosen from the set of the testing criteria presented by Souza [10] and from the criteria used by the functional testing technique suggested by Carniello [1].

Although promising, the criteria proposed in these studies have not been validated in real database project. Thus, in this paper, we report an experiment using these criteria in a database project of an Information Technology (IT) development company.

The experiment helped to demonstrate the importance of these criteria in the improvement of the database project. The chosen criteria was shown to contribute to the detection of different types of faults in the analyzed database project, enhancing the quality of applications that use the database.

Besides testing the criteria in real application, we also investigated which criteria have higher chances to contribute to the detection of specific database project faults.

Thus, this study aimed to:

- Use the RE functional testing criteria (Equivalence Class Partitioning and Boundary Value Analysis) and also the criteria generated by Souza [10] through the restrictions of the relational model based schemes (Structural Relationships, Domain Attributes, Keys, Referential Integrity, Semantic Integrity and Functional Dependency) in a database project of an IT company;
- Build and run the corresponding TC and;
- Measure the strength of each criterion, emphasizing the importance in relation to the detection of faults in the project in question.

This paper is organized as follows. Section 2 presents the main concepts and terminology in the context of software testing for database project, as well as the criteria explored in this work. Section 3 shows a case study to be explored with the functional testing criteria on database project. Section 4 presents the results obtained. Section 5 presents the conclusion and future work.

II. BACKGROUND OF FUNCTIONAL TESTING IN A DATABASE PROJECT

The database testing techniques are applied during the creation of a database application, aiming to evaluate six levels of integrity which are: the Structural Relationships, the Domain Attributes, the Keys, the Referential Integrity, the Semantic Integrity and the Functional Dependency.

 ISBN: 978-1-61208-367-4

The functional testing criteria (Equivalence Class Partitioning and Boundary Value Analysis), have been adapted to test the attributes of the tables (Domain Attributes criterion) in order to check valid and invalid values on the domains used by the project, which would result in eight testing criteria for the database project.

The following are the definitions and terminology adopted for each criteria of the functional testing in a database project. In Section A, the functional testing criteria is presented before introducing the adaptations for their use in database projects.

A. Functional Testing Technique

The functional testing is a technique used in the creation of RE in order to exercise the values of the domain of each attribute of the tables in the database project [9][14]. Thus, the functional testing contributes to the improvement of the creation of the tables and also forces the creation of checks for each of the attribute types, in order to respect the database constraints. Functional testing can be done using the Data Manipulation Language (DML) or in conjunction with the database application software [5][10].

The functional testing technique contributes to detect faults that occur when specifying the boundaries of values that can be assigned to each attribute [13]. Moreover, an elaborate specification obtained by the user in the analysis phase is critical to identify these faults.

Therefore, the functional testing criteria are based on the database specification to generate the RE which are used to produce the corresponding TC. However, the generated TC must test the criteria without affecting other database constraints that are not related to the criteria being checked [1][11].

Then, it is understood that the database specification is used both to build a program and to contribute to the generation of RE-based on the specification criteria, and subsequently indicate mechanisms for the production of TC.

In the database project, presented in the case study, we used both the Equivalence Class Partitioning and Boundary Value Analysis as criteria to test tables attributes. These criteria are presented in the following two sections.

1. Equivalence Class Partitioning Criteria

The Equivalence Class Partitioning criteria is a black box testing technique that divides the input domain (of the attributes) through the specification conditions of a given data type classes, i.e., equivalence classes, of which TC are derivative [13].

Once the equivalence classes have been established, it can be assumed, with some certainty, that any member of a class can be considered a representative of it, and every member value should behave similarly, i.e., if one member causes a fault, then any other will also cause the same fault. Thus, the criteria reduces the input domain to a passive size to be treated during the testing activities [5].

An equivalence class represents a set of valid states (expected inputs) or invalid entries (not expected) to the entry conditions, here represented by the attributes of the database tables [7].

The usage of the equivalence classes is composed by two phases: identification of equivalence classes and the generation of the corresponding TC [8].

When an input attribute of an equivalence class results in [13]:

- *Use of Intervals*: One valid and two invalid class are defined, i.e., an invalid value would be well below the lower limit and well above the upper limit;
- *Use of Specific Value*: One valid and two invalid class are defined; i.e., the value (valid) itself and a lower value, and other higher (invalid);
- *Use of an Element of a Set*: A valid class (within the set) and an invalid (outside the set) are defined;
- *Use Boolean*: A valid class (T or F) and invalid one (other than T or F) are defined.

Thus, partitioning into equivalence classes for the attributes of the tables involved, aims to produce TC who discover several classes of errors and thereby reduce the total number of TC required to satisfy the criteria [8][13].

However, this criterion can also be classified as a systematic method for the assessment of requirements, in addition to restricting the number of existing TC [3][4].

And besides that, another black box testing technique called Boundary Values Analysis criteria uses the approaches of Equivalence Class Partitioning, being seen as complementary, thus making it more systematic [13].

2. Boundary Value Analysis Criteria

The Boundary Value Analysis criteria checks more rigorously the boundaries associated with the conditions of the input attributes, i.e., exercising the boundary values [5].

And according to Myers [8], it can be said that the TC, which explores the boundary conditions, has a higher probability of finding faults. This criterion exercise the conditions of entry, and also derived the TC output to the domain when necessary [7].

The guidelines for the Boundary Value Analysis are similar to Equivalence Class Partitioning criteria as the following [13]:

- If an input condition to specify an interval determined by the values *A* and *B*, TC must be designed with values *A* and *B*, just above and just below *A* and *B* respectively;
- If an input condition specify multiple values, TC are created to exercise minimum and maximum values. Values just below and just above the minimum and maximum are tested;
- Application to output conditions, the first and second guide;
- If the internal data structures of the program have identified limits, must be projected to TC to exercise this data structure at its boundary.

Finally, if a tester apply all these guidelines, the test itself, and is more systematically, it will be complete, having a greater likelihood of fault detecting.

B. Functional Testing Specific of the Database Projects

The functional testing in a database project is to validate the specification through the DML statements, which

contribute in detecting various problems in the construction of the database project, making this type of testing becomes difficult for the following reasons [2]:

- The construction of the database to test (choice of schemes and values) involves some important and relevant factors in the generation of TC, to meet each RE of the criteria. The selection of data is essential to getting a good set of TC, since it will be the entrance to any Structured Query Language (SQL) statements;
- The applications are not just a set of statements, in preparing the database testing. Therefore, the data should be useful to the greatest possible number of instructions, for loading test data with different information for each query has a high cost;
- The information generated for testing may be modified during the execution of SQL statements. Consequently, when designing a database to assist the test, it is necessary to consider the order in which SQL statements are executed and whether they will modify the data to be input for subsequent executions with a view that relations are persistent variables;
- As in imperative languages, SQL statements can be parameterized by variables and constants and when designing the test plan, these inputs should also be considered in addition to the test data provided by the same functional testing criteria of database project;
- The adequacy of the data test unit generated it is necessary to check whether the test unit really covers all possible situations and whether the output obtained by applying the test plan satisfies the requirements for which the database has been designed in such a case.

Anyway, the functional testing of the database project involves the following steps [6]:

- Extraction of information from database schema;
- Generation of test data and filling in the database testing;
- Generation of TC as input for database;
- Validation of the state of the database and exit after execution.

To exercise the test in database project, some criteria may be used, aiming to cover different fault types.

C. Functional Testing Criteria in a Database Project

The specific criteria for database have used features exercising relations represented in the database. Thus, some criteria require the generation of TC that exercise the attributes of the same relations and other criteria that exercise the attributes of different relations, which forces the production of TC that involves a number of DML statements in one or more connections to the test.

In this work, we use the term Relation instead of Table, to keep the terminology relational database. Taking into account that the functional testing approaches consider the domain variables based on the system specification in order to work out a database application and other characteristics of the database are investigated through based criteria in: the Structural Relationships, the Domain Attributes, the Keys, the Referential Integrity, the Semantic Integrity and the Functional Dependency [10].

1. Based-on-Structural-Relationships Criterion

The Based-on-Structural-Relationships criterion has two sub-criteria that exercise multiple relations simultaneously during the test: *all-maximum-cardinality* and *all-the-minimum-cardinality*. Given two Relations *A* and *B*, then the Based-on-Structural-Relationship criterion must generate TC that exercises the cardinality relationships between *A* and *B* in order to verify their specifications [10].

Definition 1: A test data set *T* satisfies the subcriteria *all-the-maximum-cardinality* constraints of the structural relationships criteria (one-to-one, one-to-many, many-to-one and many-to-many) between two Relations *A* and *B* if the actions of cardinality between *A* and *B* are met by application.

The sub-criteria *all-maximum-cardinality* is called exercised when *T* satisfy the criteria, ensuring that:

- Relation *A* has a one-to-one relationship with Relation *B* or;
- Relation *A* has a (zero or many)-to-one relationship with Relation *B* or;
- Relation *A* has a one-to-many relationship with Relation *B* or;
- Relation *A* has a many-to-many relationship with Relation *B*.

Definition 2: A test data set *T* satisfies the sub-criteria *all-minimum-cardinality* if the structural constraints of relationship (total and partial participation) are exercised between the Relation *A* and Relation *B* and if the actions of minimum cardinality between *A* and *B* are met by the database application.

The sub-criteria *all-minimum-cardinality* is considered to exercise when *T* satisfy the criteria, ensuring that there exists at least one relationship between *A* and *B*, total or partial.

2. Based-on-Domain-Attributes Criterion

Souza [10] defined the Based-on-Domain-Attributes criterion: *all-domain-attributes*, since is the same exercising the attributes of the same relation.

Definition: A test data set *T* satisfies the sub-criteria *all-domain-attributes* if all domain constraints (Check, Data Types and Allow Nulls) of the attributes of a relation are satisfied.

The sub-criteria *all-domain-attributes* of a relation is called exercised when *T* satisfies the criteria, ensuring that the values of the attributes domain of this relation:

- Whether checked all valid and invalid conditions for each attribute, respecting its data type;
- The conditions specified in accordance with Check (valid and invalid situations in relation to clause Check) clause were satisfied;
- Comply with conditions of null or not null values established by the Allow Nulls (null or not null) clause.

 ISBN: 978-1-61208-367-4

3. Based-on-Keys Criterion

In the Based-on-Keys criterion, defined by Souza [10], there is a need for exercising existing rules in Database Management Systems (DBMS) in which all Primary Keys (PK) must be unique and not null. The sub-criteria were established: *all-primary-keys.*

Definition: A test data set *T* satisfies the sub-criteria *all-primary-keys* if all restrictions related to PK of a relation are satisfied.

The sub-criteria *all-the-primary-keys* is called exercised when *T* satisfy the criteria, ensuring that:

- Occurring uniqueness of the value of PK;
- The PK value is not null.

4. Based-on-Referential-Integrity Criterion

The Based-on-Referential-Integrity criterion has as sub-criteria for the exercise of another relation: *all-foreign-keys*. This means that the references between the relations must satisfy the constraints between non-verbal relationships of two or more relations [10].

Definition: A test data set *T* satisfies the sub-criteria *all-foreign-keys* if the referential integrity constraints, Foreign Key (FK) and relationship between relations, or a relationship between *A* and *B* are satisfied.

The sub-criteria *all-foreign-keys* is called exercised when *T* satisfy the criteria, ensuring that:

- A tuple in Relation *A*, referenced by FK, belongs to other Relation *B* be the result of an existing tuple in the relationship between relations *A* and *B*;
- A set of attributes FK in the scheme of the Relation *A* is a FK of the relationship that references the Relation *B*;
- The attributes of FK of the Relation *A* have the same domain as the attributes of the PK of Relation *B*.

5. Based-on-Semantic-Integrity Criterion

Souza [10] defined the Based-on-Semantic-Integrity criterion: *all-semantic-attributes*. Being that it exercises the actions of semantic attributes and allowed values transitions valid values are in the same relation.

Definition: A test data set *T* satisfies the sub-criteria *all-semantic-attributes* if all semantic integrity constraints (between attributes and Check of dependent attributes) of a relationship are satisfied and the dependent attributes are in the same relation.

The sub-criteria *all-semantic-attributes* is called exercised when *T* to satisfy the criteria, ensuring that:

- The value attribute of a relation satisfies the semantic condition depending if the attribute may be the same relation or in a different relation.

The Semantic Integrity is presented here as the complement of Functional Dependency when it falls on the semantics of the attribute in question.

For example, a date of birth of a parent regarding the date of birth of a descendant or the salary of an employee should not exceed the salary of the manager of the employee.

6. Based-on-Functional-Dependency Criterion

The Based-on-Functional-Dependency criterion, defined by Souza [10], exercises the attributes distinct between the same relation or different relations to which it belongs. The sub-criteria was established: *all-attributes-functionally-dependent*.

Definition: A test data set *T* satisfies the sub-criteria *all-attributes-functionally-dependent* if the restriction of functional dependency between attributes of one or more relations is satisfied.

The sub-criteria *all-attributes-functionally-dependent* is called exercised when *T* satisfy the criteria, ensuring that an attribute of a:

- Relation *B* uniquely determines another attribute of Relation *A* and actions occur in distinct dependency relations;
- Relation can also be dependent on another attribute in the same relation. This can occur whenever there is information of an attribute that are formed by the values of other attributes.

III. CASE STUDY

In partnership with Laboratory of Quality Milk (LQL) belonging to the Food Research Center, Veterinary School of the Federal University of Goiás (Universidade Federal de Goiás), at Goiânia, was developed the Panel of Quality Milk (PQL) solution, which aims to provide customers the LQL a set of milk strategic information analyzed in the laboratory, plus a knowledge base produced by researchers at the institution [12].

The goal of this solution is to encourage continuous learning and the improvement of the final quality of the Brazilian milk, leading strategic real time information to the agents of the milk chain.

For dairy, the solution helps reducing operational costs, increasing profitability and opening new markets, promoting the improvement of the quality of the purchased milk yield and production.

Moreover, the operation of the PQL is provided by information extraction from milk samples (results of analysis) were collected and sent to the laboratory as well as those identifications its (producers, farms, animals and dairy products) directly from the LQL database.

The extraction of such data is performed daily at scheduled times, forming a database constituted by historical milk testing, which will be subject to statistical analyses by the PQL tool. These analyzes are presented to dairy through a website through authenticated access.

However, the integration architecture of the system is distributed in two locations: LQL (Database and Extractor) and PQL (Database, Integrator, Controllers and Web Browsers).

Finally, the application consists of:

- Registers (*Online Help*, *Cities*, *Farms*, *Dairy*, *Paper*, *People*, *Producers*, *Fixed Price Table*, *Bonus/Punishment Table*, *Errors Types*, *Users and Milk Volume*);

 ISBN: 978-1-61208-367-4

- Settings (*Fat X CCS*, *Histograms*, *Lactose X CCS*, *Protein X CCS*, *Industrial Performance*, *Tank Volume X CCS*, *Animal Volume X CCS* and *About System*);
- Panels grouped in versions: Basic (*Collection and Recollect with a Compliance IN62*), Standard (*History of Quality*, *Producer Mirror*, *Indicators of Routes* and *Route Mirror*), Advanced (*Decision Cube*, *Errors Cube*, *Distance and Volume* and *Distance Mirror*) and Full (*Analysis of Income*, *Statement of Producer*, *Pay Per Quality* and *View Cluster*).

A. Database of PQL Project

The DBMS employed in PQL Project was PostgreSQL was due to the fact that LQL make use of it and also the large data volume that the software will behave. Moreover, this DBMS is free, high performance, highly scalable platform.

The structure of this database includes tables, fifty eight, and eight of these were selected because they are essential and relate to virtually all other tables that make up the software. They are: *analysiserror*, *analysisresult*, *baseprice*, *farm*, *routefarm*, *monthclusterfarm*, *userroles* and *person*.

After this assignment, their relationships with other tables that make up the PQL database were identified and mapped, as the following:

- *analysiserror*: *animal*, *client*, *farm* and *sampleerror*;
- *analysisresult*: *animal*, *client*, *farm*, *sampleerror*, *casein*, *cbt*, *ccs*, *esd*, *est*, *fat*, *ibc*, *proteins* and *urea*;
- *baseprice*: *dairy*, *farm* and *price*;
- *farm*: *city*, *farmer*, *milkorigin*, *person1*, *person2* and *lqlcode*;
- *routefarm*: *dairy*, *farmcode* and *route*;
- *monthclusterfarm*: *farm* and *monthcluster*;
- *userroles*: *roles* and *users*;
- *person*: *city*.

They still used the Entity Relationship Diagram (ERD) to recognize these relationships, with the intention of presenting the dependencies between tables that have gone through database functional testing with their respective domains and specificities.

A testing technique with their respective criteria was established to derive their due RE provided the generation of TC and the extraction of its expected results.

Then, a specific and isolated environment testing was structured and also given a load on database project for the tables involved in this testing activity were populated.

Upon execution of the TC, a comparison between expected results and obtained results was performed aiming to verify the effectiveness of the criteria employed in the functional testing of the database project.

B. Exploited Criteria

Among the functional testing specific of the database project criteria presented in Section 2, not all were tested. For example, it was not possible to test only Based-on-Semantic-Integrity criterion, because this characteristic was not included in the Business Plan of PQL Project.

Some examples of RE, description of TC, Inputs and Expected Results applying these criteria, which were abstracted from the document TC project of the PQL project, i.e., the test specification of the same, are shown in Table I.

TABLE I. EXAMPLES OF FUNCTIONAL TESTING CRITERIA IN A DATABASE PROJECT

Functional Criteria	RE	TC	Input	Expected Results
Based-on-Structural-Relationships	Analyze the sample result of the animal.	Modify the date of the test result of an animal.	Analysis of results of animal "57" the date "2012-05-23" to "2012-07-02".	Occurrence
		Remove an animal that has a result of analysis.	All animals that have analysis results.	Not Occurrence
Based-on-Domain-Attributes (Equivalence Class Partitioning and Boundary Value Analysis)	Specify the creation date of a farm route.	Enter a valid date in the creation of a farm route.	Date = 2011-05-05.	Occurrence
		Enter a valid null in the creation of a farm route.	Date = *null*.	Not Occurrence
Based-on Keys	Check the consistency of the PK of a farm.	Insert a single PK on a farm.	PK = 943.	Occurrence
		Insert a null PK on a farm.	PK = *null*.	Not Occurrence
Based-on-Referential-Integrity	Analyze the sample error of the animal.	Insert a parsing error for a nonexistent animal.	Error Analysis "193" for the animal nonexistent "0".	Not Occurrence
		Remove an animal that has the error analysis.	Animal which has error analysis.	Not Occurrence
Based-on-Functional-Dependency	Determine the client's name and dairy.	Modify the name of the client and dairy.	Dairy = Parmalat Brazil S/A Food Industry.	Occurrence
		Modify the name of the client and dairy to a null value.	Dairy = *null*.	Not Occurrence

The Based-on-Domain-Attributes criterion was exercised in conjunction with Equivalence Class Partitioning and Boundary Value Analysis criteria because both evaluate the

specificities of the attributes that make up the database project tables.

Thus, the Based-on-Domain-Attributes criterion was exercised along with the functional technique being Equivalence Class Partitioning and Boundary Value Analysis criteria, because of them assessing the specificity of each of the attributes that make up a database table.

The functional testing criteria of the database project, explored in this paper are shown in Table II.

TABLE II. FUNCTIONAL TESTING CRITERIA IN A DATABASE PROJECT

Functional Criteria	Functional Subcriteria	TC Exercises
Based-on-Structural-Relationships	*all-maximum-cardinality* *all-minimum-cardinality*	Maximum Cardinality; Minimum Cardinality.
Based-on-Domain-Attributes (Equivalence Class Partitioning and Boundary Value Analysis)	*all-semantic-attributes*	Occurrence of a Domain; Allow Null Value.
Based-on-Keys	*all-primary-keys*	PK; Allow Null Value.
Based-on-Referential-Integrity	*all-foreign-keys*	FK; Permit Null Key.
Based-on-Functional-Dependency	*all-attributes-dependent-functionally*	Dependent Attribute.

Thus, other criteria database demonstrates aspects of verifying how it was built and even though the current DBMS preserve these properties, they were included only for verification.

According to the tables of the database and the established performance criteria yielded a model capable of revealing the RE needed to obtain their corresponding TC.

TABLE III. EXTRACTION MODEL OF THE FUNCTIONAL TESTING CRITERIA IN A DATABASE PROJECT

Functional Testing Criteria	
Equivalence Class Partitioning	
Condition	For each attribute of a Table, Attribute values with sequential domains: L_i until L_s.
Group 1	**RE01** – Valid value until L_s ranging from L_i (L_i <= Attribute <= L_s); **RE02** – Invalid value below the L (Attribute < L_i); **RE03** – Invalid value higher than L_i (Attribute > L_s).
Condition	For each attribute a table attribute belonging to a set of values: Attribute ∈ {a, b, c, d}.
Group 2	**RE04** – Valid within the set value (Attribute ∈ {a, b, c, d}); **RE05** – Invalid value out of the set (Attribute ∉ the set {a, b, c, d}: {e}).
Boundary Value Analysis	
Condition	For each attribute of a table, with attribute values in the domain limit L.
Group 3	**RE06** – Valid value equal to the limit of **L** (**L** = Attribute); **RE07** – Invalid value lower next to **L** (Attribute < **L**); **RE08** – Invalid value near the top **L** (Attribute > **L**).
Based-on-Domain-Attributes	
Condition	For each attribute of a table, the field mapping: *Data Type* and *Allow Nulls*.
Group 4	**RE09** – Data Type (Attribute of type *Numeric*); **RE10** – Data type (Attribute of type *Date/Time*); **RE11** – Data type (Attribute of type *String*); **RE12** – Allow Nulls (Attribute *Null*); **RE13** – Allow Nulls (Attribute *Not Null*).
Based-on-Keys	
Condition	For each key of a table, the mappings of keys: *PK*.
Group 5	**RE14** – PK (Candidate Key Simple); **RE15** – PK (Candidate Key Composite).
Based-on-Structural-Relationship	
Condition	For each ratio of a table, the mappings of relations: *Relationship*, *Cardinality* and *Dependence*.
Group 6	**RE16** – Relations (Relations Association and Dependence); **RE17** – Cardinality (Relationship of Cardinality 1 – 1); **RE18** – Cardinality (Relationship of Cardinality 1 – N); **RE19** – Cardinality (Relationship of Cardinality N – N); **RE20** – Dependence (Specialization); **RE21** – Dependence (Generalization);
Based-on-Referencial-Integrity	
Condition	For each key of a table, the mappings of keys: *FK*.
Group 7	**RE22** – FK (Relationship cardinality); **RE23** – FK (Dependence).
Based-on-Functional-Dependency	
Condition	For each attribute of a table, the field mapping: *Check*.
Group 8	**RE24** – Check (Extend Relationship).

Therefore, the organization of these criteria is as shown in Table III, considering the specificities of both functional testing criteria as the for database criteria.

IV. OBTAINED RESULTS

Results for functional testing criteria in a database project used in this study were obtained through test analysis based on the coverage percentage for the quantity of RE exercised by the TC.

Altogether, there were 443 RE, generating 425 TC is needed in this step. Therefore, it was found that all the TC has been run and the database project also acquired that is a 100% coverage for the criteria.

Still, were achieved the results of the functional testing criteria (Equivalence Class Partitioning and Boundary Value Analysis) employees in Based-on-Domain-Attributes criterion. Therefore, all TC related in the criteria were executed and, furthermore, achieved a 100% coverage.

The results stemmed from the implementation of a specific functional testing in a database project through the exercise of the analyzed criteria to be presented in Table IV.

TABLE IV. RESULTS OF THE TEST RUN

Functional Testing Criteria	RE	TC	Defects
Based-on-Domain-Attributes (Equivalence Class Partitioning and Boundary Value Analysis)	171	171	9
Based-on-Keys	28	21	0
Based-on-Structural-Relationships	124	113	0
Based-on-Referential-Integrity	100	100	0
Based-on-Functional-Dependency	20	20	0
Grand Total	**443**	**425**	**9**

In general, it is observed that the other specific criteria of database testing help verify that the project meets specified correctly to ensure a good quality of the generated information.

The types of fields *date*, *number*, *text*, *email* and *website* were verified through Based-on-Domain-Attributes criterion along with functional testing criteria Equivalence Class

 ISBN: 978-1-61208-367-4

Partitioning and Boundary Value Analysis, which consequently showed the faults in a database project.

Finally, the Boundary Value Analysis criteria detected most of the faults identified in the database project because a large amount of these faults is in the limits of the domains of its attributes.

V. CONCLUSION AND FUTURE WORK

According to functional testing criteria on a database project, there was a high efficiency degree in detection of faults during the execution of TC for the RE generated in PQL's Reports. All functional testing criteria applied reached a coverage of 100% in relation the RE.

Nine faults were detected over the following criteria, Equivalence Class Partitioning (three faults) and Boundary Value Analysis (six faults) together with Based-on-Domain-Attributes criterion.

The Equivalence Class Partitioning criteria contributed to the definition of TC by the class of faults, reducing the total number of TC generated in the PQL's Project. The Boundary Value Analysis criteria allowed to observe most faults tend to occur at the borders of the domain.

Furthermore, because the Based-on-Keys, Based-on-Structural-Relationships, Based-on-Referential-Integrity and Based-on-Functional-Dependency criterion contributing with corrections in existing restrictions on a project database, so no fault was detected during the execution of the TC. The Based-on-Semantic-Integrity criterion was not used in this project considering that the documentation does not make any reference to these dependencies.

Finally, the combination of Based-on-Domain-Attributes criterion also provides Equivalence Class Partitioning and Boundary Value Analysis criteria by exploring a high level of faults detection, just treating the specific attributes. And thus, can be utilized in the database project.

For future work, an object of study with the purpose of applying the criteria presented is being constructed, such as the Based-on-Functional-Dependency and Based-Semantic-Integrity. These criteria can improve the detection of failures a project database.

REFERENCES

[1] A. Carniello, Test Structure Based on Use-Case, FEEC/UNICAMP, Campinas, 2003.

[2] A. D. Suarez, A. S. Simão, J. C. Maldonado, and P. C. Masiero, "Using an SQL Coverage Measurement for Testing Database Applications", In: ACM SIGSOFT Software Engineering Notes, New York, pp. 253-262, 2004.

[3] A. L. Domingues, Assessment Criteria and Test Tools for OO Programs, ICMC/USP, São Carlos, 2002.

[4] A. M. Vincenzi, E. F. Barbosa, J. C. Maldonado, M. E. Delamaro, M. Jino, and S. R. S. Souza, Introduction to Software Testing, Teaching Notes, ICMC-USP, São Carlos, 2004.

[5] A. M. Vincenzi, J. C. Maldonado, and S. C. Fabbri, Introduction to Software Testing: Functional Testing, Rio de Janeiro: Elsevier, 2007.

[6] D. Chays, E. J. Weyuker, F. I. Vokolos, P. G. Frankl, and S. Dan, "A Framework for Testing Database Applications", In Proc. of the ACM SIGSOFT Intl. Symp. On Software Testing and Analysis, Vol. 25 Issue 5, August 2000, pp. 49-59.

[7] G. J. Myers, Software Reliability Principles and Practices, 1st ed. New York: John Wiley & Sons, INC., p. 360, 1976.

[8] G. J. Myers, T. Badgett, and T. M. Thomas, The Art of Software Testing. 2nd ed., New York: John Wiley & Sons, INC., 2004.

[9] I. Burnstein, Pratical Software Testing: A Process Oriented Approach, New York: Springer-Verlag, p. 709, 2002.

[10] J. P. Souza, Functional Testing Application DB Based on UML diagram, UNIVEM, Marília, 2008.

[11] J. Tian, Software Quality Engineering, Texas: John Wiley & Sons, INC., p. 412, 2005.

[12] Milk Panel, http://www.paineldoleite.com.br/site/ Aug/Sept 2014.

[13]. S. Pressman, Software Engineering, 6th ed., São Paulo: McGraw-Hill, p. 720, 2006.

[14] W. E. Lewis, Software Testing and Continuous Quality Improvement, 2nd ed., Florida: CRC Press LLC, p. 534, 2004.

 ISBN: 978-1-61208-367-4