Studying The Free Database Protection From the inputting wrong data
And undesirable Information

م.م. وليد خالد سليمان شعبان
جامعة تكريت/ كلية علوم الحاسوب والرياضيات

Introduction

Data collection has become a ubiquitous function of large organizations (not only for record keeping, but to support a variety of data analysis tasks that are critical to the organizational mission. Data analysis typically drives decision–making processes and in an increasing number of settings is the raison d'etre of entire agencies. Despite the importance of data collection and analysis, data quality remains a pervasive and thorny problem in almost every large organization. The presence of incorrect or inconsistent data can significantly distort the results of analyses, often negating the potential benefits of information–driven approaches. As a result, there has been a variety of research over the last decades on various aspects of data cleaning: computational procedures to automatically or semi–automatically identify { and, when possible, correct { errors in large data sets. In this report, we survey data cleaning methods that focus on errors in quantitative attributes of large databases, though we also provide references to data cleaning methods for other types of attributes. The discussion is targeted at computer practitioners who manage large databases of quantitative information, and designers developing data entry and auditing tools for end users.[4] The objective of each business application is data processing. Some input is processed by some rules given by the program and this generates some output. If a hacker sends input to the application which is not as well formed as expected than the input can be processed incorrectly and reveal protected resources. The classical firewall can defend your application from attacks performed at lower OSI layers but it can not protect against attacks by the use of malicious input. Thus there is a need to protect your application from invalid input. Input validation is effective because this countermeasure covers several of the most common web application vulnerabilities. According to the OWASP list of the Top Ten Most Critical Web Application Security Vulnerabilities1 the items “A1 – Invalidated Input”, “A5 – Buffer Overflows”, and “A6 – Injection Flaws” can be mitigated –by input validation. Input validation is no new invention of security experts. It has been a best practice from the beginnings of programming to verify all data syntactically. Unfortunately in the times of “rapid prototyping” and “time to market” pressure this guideline has been lost widely. Even some modern development environments does not support input validation in a convincing way. Thus programmers have to find their own way of input

92
validation From a scientific point of view, input validation is a semantic problem. Input will be accepted and processed by a program. The input is an element of an object language. The program itself uses a meta language (script code, assembler code, etc) to control the processing. The objective of an attacker is to deliver object language which partly will be interpreted as meta language. We discuss several input validation solutions and present best practices and basic examples in Microsoft’s ASP.NET and in Java’s struts framework.[5]

**Data Types in Access database**

We get the Data type in Access, we must assign a data type for each field. The data type determines what values you can enter for a field and what other properties the field will have. we want to choose a data type that most closely matches the kind of data in plan on storing in that field, especially in larger databases. This will help keep file sizes manageable and will optimize Access’s performance. The data types are as follows:

- **Text**: Use for field values that contain letters, digits, spaces, and special characters. Text fields can contain up to 255 characters. The default field size for a text data type is 50.

- **Memo**: Use for long text comments. Memo fields can contain up to 64,000 characters. Unlike text fields, however, you cannot search or sort records based on the contents of a memo field.

- **Number**: Use for numeric values. Because number fields are more restrictive than text fields, only chose this data type for numerical entries that will be used in calculations (e.g., percentages, quantities). Do not use this field type for currency values (see Currency, below). The default field size for a number data type is Long Integer. For more information about field sizes for number data types, press the F1 key on your keyboard when you are clicked in the Field Size property box.

- **Date/Time**: Use for dates and times. Date/time fields have a number of standard display options, but can also be customized to display in other formats. Access recognizes years as 4-digit numbers, even if only the final two digits are displayed.

- **Currency**: Use for currency values. Currency fields are similar to the number data type, except that the decimal places and field size are predetermined, and calculations performed using the currency data type are not subject to round-off error.

- **AutoNumber**: Use this when you want integers or a value automatically inserted in the field as each new record is created. You can specify sequential numbering or random numbering. Using AutoNumber guarantees a unique field value, which can serve as a table’s primary key. AutoNumber fields cannot be edited.
• Yes/No: Use this data type for fields that indicate the presence or absence of a condition (e.g., such as whether a person has enrolled, if their application has been submitted, etc.) Yes/No fields store a numeric value and can display a numeric value, text, or a graphic “check box.” The value –1 means “yes”, “true”, or “on” and 0 means “no”, “false”, or “off.”

• OLE Object: Use for data or files that are created in other software applications, such as photographs, video images, graphics, drawings, sound recordings, spreadsheets, word processing documents, etc. [3]

Sources of Error in Data

Before a data item ends up in a database, it typically passes through a number of steps involving both human interaction and computation. Data errors can creep in at every step of the process from initial data acquisition to archival storage. An understanding of the sources of data errors can be useful both in designing data collection and duration techniques that mitigate the introduction of errors, and in developing appropriate post–hoc data cleaning techniques to detect and ameliorate errors. Many of the sources of error in databases fall into one or more of the following categories:

1- Data entry errors: It remains common in many settings for data entry to be done by humans, who typically extract information from speech (e.g., in telephone call centers) or by keying in data from written or printed sources. In these settings, data is often corrupted at entry time by typographic errors or misunderstanding of the data source. Another very common reason that humans enter ‘dirty’ data into forms is to provide what we call spurious integrity many forms require certain fields to be filled out, and when a data–entry user does not have access to values for one of those fields, they will often invent a default value that is easy to type, or that seems to them to be a typical value. This often passes the crude data integrity tests of the data entry system, while leaving no trace in the database that the data is in fact meaningless or misleading.

2- Measurement errors: In many cases data is intended to measure some physical process in the world: the speed of a vehicle, the size of a population, the growth of an economy, etc. In some cases these measurements are undertaken by human processes that can have errors in their design (e.g., improper surveys or sampling strategies) and execution (e.g., misuse of instruments). In the measurement of physical properties, the increasing proliferation of sensor technology has led to large volumes of data that is never manipulated via human intervention. While this avoids various human errors in data acquisition and entry, data errors are still quite common: the human design of a
sensor deployment (e.g., selection and placement of sensors) often affects data quality, and many sensors are subject to errors including miscalibration and interference from unintended signals.

3- Distillation errors: In many settings, raw data are preprocessed and summarized before they are entered into a database. This data distillation is done for a variety of reasons to reduce the complexity or noise in the raw data (e.g., many sensors perform smoothing in their hardware), to perform domain-specific statistical analyses not understood by the database manager, to emphasize aggregate properties of the raw data (often with some editorial bias), and in some cases simply to reduce the volume of data being stored. All these processes have the potential to produce errors in the distilled data, or in the way that the distillation technique interacts with the final analysis.

4- Data integration errors: It is actually quite rare for a database of significant size and age to contain data from a single source, collected and entered in the same way over time. In almost all settings, a database contains information collected from multiple sources via multiple methods over time. Moreover, in practice many databases evolve by merging in other pre-existing databases; this merging task almost always requires some attempt to resolve inconsistencies across the databases involving data representations, units, measurement periods, and so on. Any procedure that integrates data from multiple sources can lead to errors.[4]

**No input validation**

In general it is not recommended to skip validation. There is an exception of this rule. If the design ensures that the input has been successfully validated before then it is acceptable to skip the input validation. However as a basic principle each component should be able to defend itself. A component may skip input validation only if it has a trust in the correctness of the input validation of the delivering component. This is a classical example of the design by contract methodology.

Enter has preconditions which must be satisfied by the caller. In this case the precondition would be “all input delivered to the called function has been successfully validated by the calling function”. This assumption must hold during over the complete software (component) lifecycle. In particular

reuse of components is susceptible for the violation of mutual contracts. (This One of the most famous examples is the crash of the first Ariane 5 rocket in 1996. A control system which was designed for Ariane 4 expected only a limited range of input data. Ariane 5 is much taller and heavier and the input data infringed the former “reasonable” assumptions.
1. The change management in the software (component) lifecycle must guarantee that all changes of the contract must be inherited to all contracts with dependable modules.

2. The validating function must “know” what malicious input is for all subsequent callable functions.

3. New callable functions in the backend can require changes in the input validation of the calling functions.

   These are three strong requirements which are difficult to assure over long times. Furthermore such strong requirements between functional unrelated modules are undesirably because they make difficult the decoupling of components.

   Best practice: It is not recommended to skip input validation at interfaces to untrustworthy sources. No input validation is acceptable for internal function and methods which can trust their callers. In particular private methods can expect trustworthy input which may not be validated again. [2]

**Validation of data**

Data Validation generally can be defined to mean: A systematic process that compares a body of data to the requirements in a set of documented acceptance criteria. The process of verification of data, either during the writing instruction program or the design of tables, fields, and so the good designer of the database management program Hoalchks does not give opportunity to the system user that mistake mistakes irrational during the process of entering data into database tables without running the system to alert the user to this mistakes and ensure full of archived data tables. Example data entry when a particular field of history to accept the wrong database management system date February 30 or until 29 February and this is a mis–often based on calendar year. Data Verification is the process of evaluating the completeness, correctness, and conformance/compliance of a specific data set against the method, procedural, or contractual requirements. [1]

**validation rule**

Data validation can be performed by several different entities. Commonly, the analytical laboratory providing the analytical data may do a limited scope data validation. The laboratory’s role is generally limited to assigning data qualifiers based on events relating to the analysis of the samples and the requirements in the methods and procedures that were utilized by the laboratory. In addition, the laboratory may follow other applicable contract requirements for data reporting and reporting. Validation of data by a party independent of the laboratory is most often the more desirable option. A full scope data validation process
performed by such a third party for the data user ensures an unbiased and technically sound outcome. In a database it is important that data is put in accurately. One method of doing that is to use validation rules to restrict what people can put in. So you might have a school where children have to be over a certain age before they can join. You could build a validation rule to ensure that requirement or to ensure that the person entering the data does not make a mistake when entering it. A validation rule will stop them making an error and with the Validation Text, it will give them a message telling them what their error is.[1],[8]

Integration of the database

Estimated to maintain the integrity of the data to remain in the data tables is accurate and that under no circumstances Zervmn future, an example of an integrative database called a record is deleted from a table, the students that you need to delete all the degrees of student grades from the tables.[6]

Protect Data Using Blacklisting

The term “blacklisting” means to filter evil input only. Unfortunately “evil input” can not be exactly defined. Usually blacklisting is based on the detection of possibly or probably or certain malicious input. The most common example is filtering of the JavaScript tag <script> to avoid Cross Site Scripting. This inhibits code injections like

<script>alert (“You have been hacked!”</script>.

Blacklisting does not enjoy a good reputation. There are some inevitable problems which have no satisfying solution.

1. One can not define all evil input. Only known attacks can be described in patterns. New attacks must be detected, analyzed, described in corresponding patterns, and afterwards implemented. This is the same awkward position of the antivirus industry: “You can only defeat what you know.”

2. A famous motto of Perl programmers is “There’s more than one way to do it.” { Perl Docs. Tbd} This holds for the bad guys too. They a very clever to develop circumventions of blacklists. For example an ASP.NET page should throw an exception if an input form field is filled with "<script>". In ASP.NET 1.1 this doesn’t hold for "<%00script>" but browser render both strings in the same matter such that an attacker can use <%00script> to perform a XSS attack. Nevertheless, blacklisting is very popular. It can partly defend against specific known threats. Sometimes it is the only possibility. If your application server has a well known but not yet patched vulnerability then blacklisting the attack at your bastion host can be an appropriate countermeasure.
Some vendors have implemented default blacklists. For example in ASP.NET all requests are checked with a hard coded list of malicious strings. {As long as the Validate Request attribute is active} Obviously this concept is deficient by design but it could be the starting point of a "poor man’s blacklist". Sometimes Blacklisting is an appropriate approach. It can be used to fill the gap between the release of a so called “Zero Day Exploit” {“Zero Day Exploits” are vulnerabilities which can are published simultaneously with an attack routine. In general an appropriate vendor patch or workaround is not available immediately} and the release of a well tested patch. Often such an exploit is identified by a characteristic attack pattern which can filter. This is not a solution for ages because later exploits can and will vary. However blacklisting can be a countermeasure to gain time for your developers fixing the real problem. A similar motivation for blacklisting is defer patching to a regular maintenance slot.

The Whitelisting is the recommended best practice for input validation. All input may only pass if it is validated as known good input. This is also called positive input validation in contrary to the negative validation of the blacklisting. This is strong requirement and in real world application not easy to implemented. The simplistic standard example is the positive validation of a ZIP with a plain regular expression. Example: the regular expression \d{5}$ can be used to validate a German zip code consisting of five digits.

Sometimes an implemented blacklisting will be used as a replacement for a whitelisting. Only for the sake of completeness we remark that you should consider blacklisting only as a workaround of inferior quality.

Best practice: Avoid blacklisting if you can do better. Whitelisting is preferable from the viewpoint of security. However blacklisting is much better than no input validation. [6]

Inputting Data methods

Major sources of conventional wisdom used to enter data into databases is the keyboard and mouse, floppy disks and optical disks, etc., but because of the tremendous development in computer systems and database management systems, which corresponds to piece development in the field of Hardware has become a new ways and non-traditional data entry to sites storage of key data in databases, and are these methods local networks and the Internet which can transfer and exchange of vast amounts of data between a large number of local databases through data systems are distributed, but for such significant disadvantages which can enter the data is contaminated and unhealthy data sources which are used in decision-making through the snooping, viruses, and some fuzzy data, and is encouraged by Elgar.
On the other hand, the quality and efficiency of the sources of moderation which is used to transfer and data entry with the vast amount of each that pay programmers and manufacturers to join forces together in order to access the system control of the strong can access to verify the optimization of data before and after the entering and transfer of sources of computers and databases.

**DATA VERIFICATION/VALIDATION IN THE PROJECT LIFE CYCLE**

EPA’s Quality System has been described in other documents issued by the EPA Quality Staff – see, for instance, EPA Requirements for Quality Management Plans (QA/R-2) (EPA, 2001a). This system provides an integrated set of policies, programs, and project-level tools, all with the common goal of producing defensible products and decisions. As shown in Figure (1), data verification...
and data validation fit into the category of project-level tools. This category of tools includes systematic project planning, project implementation in the field and analytical laboratory, and the assessment phase, where data are evaluated and prepared for use.

Figure 2 illustrates the overall framework and feedback loops that may be needed for data verification and data validation. Although data verification and data validation are both considered assessment tools, chronologically they occur prior to the formal data quality assessment (DQA) process. DQA is described in the Guidance for Data Quality Assessment: Practical Methods for Data Analysis (QA/G-9) (EPA, 200b). As discussed in subsequent chapters, the goal of data verification is to ensure and document that the data are what they purport to be, that is, that the reported results reflect what was actually done. Data validation is generally carried out (usually by an external party) as part of the assessment phase. The goal of data validation is to evaluate whether the data quality goals established during the planning phase have been achieved. As shown in Figure (2),

data validation involves the outputs of the planning and implementation phases. The data validator may also be requested to perform a detailed investigation of particular data records that need special interpretation or review, referred to as a focused data validation[7].

Figure 2. Data Verification and Data Validation Components in the Project Life Cycle
Some Validation Data Application

1–ASP.NET Validator Controls

Microsoft ASP.NET contains very powerful validators for input validation which can be easily integrated in their development environment Visual Studio 2005. In particular, the Regular Expression Validator is a useful tool from the viewpoint of security.

```
<asp: Regular Expression Validator
   ID="RegularExpressionValidator1"
   Run at="server"
   Control To Validate="txt User Name"
   Error Message="Username is not valid."
   Validation Expression="\w{1,10}"
/>
</asp: Regular Expression Validator>
```

Example 1: Input validation in ASP.NET using the Regular Expression Validator control

The validation expression in this example allows all usernames with one up to ten word characters ("\w"). There is a pitfall while using ASP.NET validators. The binding of a validator control to a form control does not change the execution flow of the page at the server side. The only effect is the setting of the "IsValid" property of the respective validation control. The developer must query this property to control the execution of the page. Additionally the validation of the client side JavaScript code can pretend a secure validation while the server side validation is still missing.

ASP.NET validation controls validate input mandatory at the server side but they can generate additional client side JavaScript validation code. Unfortunately, the interpretation of regular expressions in JavaScript is slightly different from the interpretation in more powerful regular expression server machines. In particular, internationalization and Unicode causes serious troubles. We recommend to tailor the validation by client side JavaScript manually in complex situations.[6]

2–ALTER TABLE – SQL Command


```
ALTER TABLE TableName1 ADD | ALTER [COLUMN] FieldName1
   FieldType [ (nFieldWidth [, nPrecision])] [NULL | NOT NULL]
   [CHECK iExpression1] [ERROR cMessageText1]]
   [AUTOINC [NEXTVALUE NextValue [STEP StepValue]]] [DEFAULT eExpression1]
   [PRIMARY KEY | UNIQUE [COLLATE cCollateSequence]]
   [REFERENCES TableName2 [TAG TagName1]] [NOCPTRANS] [NOVALIDATE]
```
The words that color red is represent the validation in SQL command using alter table Command. To description the parameter detail in the following statement NULL | NOT NULL

Specifies whether null values are allowed in the field. NULL permits null values, while NOT NULL does not allow null values. If one or more fields can contain null values, the maximum number of fields the table can contain is reduced from 255 to 254.

CHECK [Expression]

Specifies a validation rule for the field. The [Expression] parameter must evaluate to a logical expression and can be a user–defined function or a stored procedure. Visual FoxPro checks the validation rule specified in the CHECK clause when a blank record is appended.

ERROR cMessageText1

Specifies an error message. Visual FoxPro displays this message when the validation rule specified with the CHECK clause generates an error. The message displays only when data is changed within a Browse window or Edit window.

AUTOINC [NEXTVALUE NextValue [STEP StepValue]]

Enables automatic incrementing for the field. NextValue specifies the start value and can be a positive or a negative integer value ranging from 2,147,483,647 to -2,147,483,647. The default value is 1. You can set NextValue using the Next Value spin box in Fields tab of the Table Designer.

StepValue specifies the increment value for the field and can be a positive, nonzero integer value ranging from 1 to 255. The default value is 1. You can set StepValue using the Step spin box in the Fields tab of the Table Designer.

Hear Same Example Applying This Parameter:

Example 1 adds a field called Fax to a Customer table and allows the field to have null values.

Example 2 makes the Cust_id field the primary key of the Customer table.

Example 3 adds a field validation rule to the Quantity field of the Orders table so that values in the Quantity field must be non-negative.

Example 4 adds a one–to–many persistent relation between the Customer and Orders tables based on the primary key Cust_id in the Customer table and a new foreign key index Cust_id in the Orders table.

Example 5 removes the field validation rule from the Quantity field in the Orders table.

Example 6 removes the persistent relation between the Customer and Orders tables, but keeps the Cust_id index tag in the Orders table.

Example 7 adds a field called Fax2 to the Customer table and prevents the field from containing null values. The new structure of the table is displayed. Two ALTER COLUMN
clauses are used to allow the field to have null values and set the default value for the field to the null value. Note that multiple ALTER COLUMN clauses are required to change more than one property of a field in a single ALTER TABLE command. The new field is then removed from the table to restore the table to its original state.

* Example 1
  
  ```sql
  SET PATH TO (HOME (2) + 'Data') && Sets path to table.
  ```

  ```sql
  ALTER TABLE Customer ADD COLUMN Fax c (20) NULL
  ```

* Example 2

  ```sql
  ALTER TABLE Customer ADD PRIMARY KEY Cust_id TAG Cust_id
  ```

  ```sql
  ALTER TABLE Customer ALTER COLUMN Cust_id c (5) PRIMARY KEY
  ```

* Example 3

  ```sql
  Alter table orders;
  ```

  ```sql
  alter column quantity set check quantity >= 0;
  ```

  ```sql
  error 'quantities must be non-negative'
  ```

* Example 4

  ```sql
  Alter table orders;
  ```

  ```sql
  add foreign key cust_id tag cust_id references customer
  ```

* Example 5

  ```sql
  Alter table orders alter column quantity drop check
  ```

* Example 6

  ```sql
  ALTER TABLE Orders DROP FOREIGN KEY TAG Cust_id SAVE
  ```

* Example 7

  ```sql
  Clear
  ```

  ```sql
  Alter table customer add column fax2 c (20) not null
  ```

  ```sql
  Display structure
  ```

  ```sql
  Alter table customer;
  ```

  ```sql
  alter column fax2 null;
  ```

  ```sql
  alter column fax2 set default. null.
  ```

  ```sql
  Alter table customer drop column fax2
  ```

The following example uses NOCPTRANS to prevent translation to a different code page. The example creates a table named 'mytable' that contains two character fields and two memo fields. The second character field, 'char2'; and the second memo field, 'memo2'; include NOCPTRANS to prevent translation.

```sql
CREATE TABLE mytable (char1 C (10), char2 C (10) NOCPTRANS;
  memo1 M, memo2 M NOCPTRANS)
```
3–Excel Data Validation Messages to Help Users

You can can be displayed when the cell with data validation is selected.

An Error Alert display messages to give instructions to the people who use your spreadsheet.

An Input Message can be displayed if invalid data is entered. You can turn this off, to allow people to enter invalid data. For example, if the data validation cell contains a dropdown list, turn off the Error Alert to allow users to type items that are not in the list.

Input Message

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>Item</td>
</tr>
<tr>
<td></td>
<td>Select a category from the dropdown list</td>
</tr>
</tbody>
</table>

4– One method of creating a check constraint is to do it when a table is created. Here is a simple CREATE TABLE script that creates a single check constraint in SQL in VFP (Visual FoxPro):

```
CREATE TABLE dbo. Payroll
(ID int PRIMARY KEY,
 Position ID INT,
 Salary Type nvarchar (10),
 Salary decimal (9,2)
CHECK (Salary < 150000.00));
```

Here I have named my check constraint CK–Payroll–Salary.

```
CREATE TABLE dbo. Payroll
(ID int PRIMARY KEY,
 Position ID INT,
 Salary Type nvarchar (10),
 Salary decimal (9,2)
CONSTRAINT CK–Payroll–Salary CHECK (Salary < 150000.00));
```

Results and Discussion

Through research on the mechanism for data entry free to the database systems has been reached over the serious and significant loss of time, effort as well as economic loss caused by the input bad data in the treatment systems, and data analysis systems such as
spreadsheets and spreadsheets systems text editor and control systems which are data used to the primary source in the work environment, automatic control and through which decisions are made, especially facilities of industrial and especially the search data within the database management systems, which controls the management systems as a payroll, personnel and student affairs, etc., as well as database management systems in servers, Internet networks and local regulations Distributed by working on the management of banks and stock markets and economic data and e–government.

The multiple ports to access the data to the PC requires a strong need for the presence or availability of systems to verify the data in the necks of the main entry gates to prevent and analyze the data before entry into the data sources, a valuable database systems. A diagnosis of errors that occur and the reasons, by the following:

1 – tremendous speed in the flood the market with systems that do not rely on measures of verification of the global data.
2 – the implementation of systems hydrocele databases without the test results and match them with legacy systems.
3 – the misuse of these systems Fbl people who have no experience necessary qualifying.
4 – the large volume of data and the unexpected with the limited database management systems working.
5 – not the protection of the data found in sites of free entry and update networks.
6 – multiple ports for data entry and increase the speed of access, which led to the dumping of data sources and data loss in the Born.
7 – there is no hiding the data sources (tables) of the system user.
8 – to rely on the system user to verify the data.
9 – the absence of analysis of database systems.

This mis–generated errors increased significantly depending on the amount of data and the conditions of introduction and methods of introduction and that we can deduce the following:

1 – place all database management systems to verify that measures have been identified in this research before the final implementation of these systems.
2 – Building systems of data protection in the free sites to receive and send data across networks.
3 – Use of modern methods of verification used by programs in the visual error diagnosis and treatment of self–validation rules depending on the appropriate database system.
4 – Avoid dumping the data sources and data loss .
5 – to subject the persons responsible for data entry to intensive courses on how to avoid mistakes and tell the programmers.
Refrains


