Poor Data Injector: Fault Model

Nuno Laranjeiro*, Seyma Soydemir* and Jorge Bernardino*†

*CISUC, Department of Informatics Engineering
University of Coimbra, Portugal
cnl@dei.uc.pt, seyma@student.dei.uc.pt
†Polytechnic of Coimbra, Portugal
jorge@isec.pt

1 Introduction

This document presents the Fault Model used with **Poor Data Injector**. Poor Data Injector is a tool that mutates the original data coming from database, according to a predefined set of mutations (i.e., this fault model). Instead of the real data, the application will be fed with mutated data (poor data quality). With this approach, Poor Data Injector tests the application and uncovers software flaws in the web application, middleware, and libraries.

The Fault Model, which specifies in detail the mutations that can occur according to the data type being used, is presented in the following tables. Below is a brief description of each table.

- **JDBC Data Grouping:** The groups of data types that will be the basis for the mutation tables are presented in Table 1. These groups were obtained by taking into account the native data types of JDBC and Java, and seeing how they interlink in order to create broader groups of data types of the same class. The need for this table arose when creating Poor Data Injector in Java, as a way to connect JDBC and Java (the language of the mutations).

- **String Data Mutations:** The mutations for the String data type are presented in Table 2, where the string *"John Smith"* was used as the example input.

- **Integer Data Mutations:** Table 3 shows the mutations defined for the integer data types. The original input used for the "Examples" column was *123*.

- **Time Data Mutations:** Table 4 presents the mutation for the Time data types. The original input used for the "Examples" column was *02:10:08*.

It should be noted that values higher or lower than the limit of the fields (hours, minutes, seconds) are considered valid in these mutations.

- **Date Data Fault Model** Table 5 presents the mutation for the Date data types. The original input used for the "Examples" column was *2020-05-10*. As happened with the Time data types, values higher or lower than the limit of the fields (year, month, day) are considered valid in these mutations.

- **Timestamp Data Mutations:** Table 6 presents the mutation for the Timestamp data types. The original input used for the "Examples" column was *2020-05-10 11:11:11.121212121*. Once again, values higher or lower than the limit of the fields are considered valid in these mutations.

- **Boolean Data Mutations:** Table 7 presents the mutation for the Boolean data types. The original input used for the "Examples" column was *True*.

- **Decimal Data Mutations:** Table 8 presents the mutation for the Decimal data types. The original input used for the "Examples" column was *123,13*.

- **Double Data Mutations:** Table 9 presents the mutation for the Double data types. The original input used for the "Examples" column was *123,13*.

- **Binary Data Mutations:** Table 10 presents the mutation for the Binary data types. The original input used for the "Examples" column was *[10, 4, 5, 1, 149]*, where each position inside the array represent the value of a byte (from 0 to 255).

- **Object Data Mutations:** Table 11 presents the mutation for the Object data types. Due to the abstract definition of a Java object, and how difficult it would be to represent in text, the example column is only filled in the first mutation.

- **Reference Data Mutations:** Table 12 presents the mutation for the Reference data types. Due to the abstract definition of a Java Reference, and how difficult it would be to represent in text, the example column is only filled in the first mutation.

2 Data Types

The glue that provides the interface between a Java program and the DBMS is called JDBC (Java Database Connectivity). Despite the relatively high amount of different data types supported by JDBC, most of them can be aggregated in general groups (e.g., TINYINT, SMALLINT, INTEGER can be grouped into Integer). After comparing the data types available in JDBC [1] and Java we reached the following mapping (Table 1):

Table 1: Mapping between our self-defined groups, JDBC types and Java types

Group	**JDBC Type**	**Java Type**
String	CHAR	String
	VARCHAR	
	LONGVARCHAR	
	CLOB	Clob
	DATALINK	java.net.URL
Decimal	NUMERIC	java.math.BigDecimal
	DECIMAL	
Boolean	BIT	boolean
	BOOLEAN	
Integer	TINYINT	byte
	SMALLINT	short
	INTEGER	int
	BIGINT	long
Double	REAL	float
	FLOAT	double
	DOUBLE	double
Binary	BINARY	byte[]
	VARBINARY	
	LONGVARBINARY	
	BLOB	Blob
Date	DATE	java.sql.Date
Time	TIME	java.sql.Time
Timestamp	TIMESTAMP	java.sql.Timestamp
Object	DISTINCT	(Depends on Object)
	JAVA OBJECT	
Reference	REF	Ref

Overall, there are 11 different groups of data types and each group can aggregate more than 1 JDBC or Java data type. Sometimes different JDBC types translate to the same type in Java. For each one of these 11 groups, a table with the applicable mutations was built, and the implementation of those mutations was done in Java code.

3 String Data Mutations

Table 2: Mutations for String data types

No	Fault Description	Argument	Configuration	Example
1	Replace by null value	-	-	null
2	Replace by empty string	-	-	“”
3	Replace by same size string	-	-	Aks9DLM34q
4	Add random nonprintable characters to string	Position	Prefix	\u0010John Smith
			Middle (Random position)	John\u0010 Smith
			Suffix	John Smith\u0010
		Length	Random between 1 and string's original size	-
5	Replace by same size alphanumeric string	Number of characters	Random quantity of characters between 1 and the total number of alphabetic characters minus 1. Randomly distributed among the available position	Jo3n Smixh
6	Add random numeric characters to string	Position	Prefix	0158John Smith
			Middle (Random position)	John01 S4mi5th
			Suffix	John Smith544
		Length	Random between 1 and string's original size - string's maximum size	-
7	Convert alphabetic characters to opposite case	Position	First character	john Smith
			Last character	John SmitH
			Random position	John sMITH
			First character of each word	john smith
			Last character of each word	JohN SmitH
			Random position in each word	JOHn sMith
		Length	Random between 1 and string or word original size minus Position	-
8	Insert whitespace	Position	Leading	John Smith
			Trailing	John Smith
			Random position between words	John Smith
		Quantity	The amount of consecutive white space characters to insert	-
9	Add characters to string	Position	Prefix	ooJohn Smith
			Middle (Random position)	John Shhhmith
			Suffix	John Smithii
		Quantity	Random between 1 and the string's original size. The new characters to add must be present in original string.	-
10	Add substring to string	Begin	From 0 to the string's original size minus 1.	-
		Length	From 1 to the string's original size minus Begin.	-
		New Position	Prefix	ohnJohn Smith
			Middle (Random position)	Johnohn Smith
			Suffix	John Smithohn
11	Move substring	Begin	From 0 to the string's original size minus 1.	-
		Length	From 1 to the string's original size minus Begin.	-
		New Position	Prefix	ohnJ Smith
			Middle (Random position)	J Sohnmith
			Suffix	J Smithohn
12	Remove substring	Begin	From 0 to the string's original size minus 1.	J Smith
		Length	From 1 to the string's original size minus Begin.	
13	Remove whitespace	Position	Leading	John Smith
			Trailing	John Smith
			Random position between words	JohnSmith
14	Delimited reverse	Begin	From 0 to the string's original size minus 2.	John Shtim
		Length	From 2 to the string's original size minus Begin	
15	Replace by predefined SQL string	-	Dictionary based: replace the string by a keyword or symbol such as SELECT, FROM, INSERT, DELETE, UPDATE, ', ",	SELECT
16	Replace with symbols	-	The string is replaced by a random symbol	$
17	Misspelling	-	Replace one word by a misspelled word (Dictionary-based) or in case of no match being found, use a random single edit operation (insertion; deletion; substution of a single character or transposition of two adjacent characters) over a randomly selected word	John Smtih
18	Replace with a limit size	Size	Java's UTF maximum size	-
			Java's UTF maximum size +1	-
19	Replace with an imprecise value	-	Dictionary based: Chooses a single random word and replaces it by the respective acronym or abbreviation	J. Smith
20	Replace by specific type string	-	Replace value with another data types (e.g., numeric, date, time, timestamp, boolean mutations, to be defined)	30
21	Replace whitespaces by	-	Replaces all the white spaces by symbols (_, . , %, $, €)	John_Smith, John%Smith
22	Add extraneous data	Position	Before	PhD John Smith
			After	John Smith CEO
23	Replace by homonyms	-	Replaces one word by its homonym, when possible.	allowed -> aloud
24	Replace by synonyms	-	Replaces one word by its synonym, when possible.	happy -> cheerful

4 Integer Data Mutations

Table 3: Mutations for Integer data types

No	Fault Description	Argument	Configuration	Example
1	Replace by random same size	-	-	145
2	Remove one random numeric character from data	-	-	13
3	Duplicate one of the numeric characters	Position	From 0 to the numeric value's original size. Repeat digit will be put in front of Position	1223
4	Invert two consecutive digits	Begin	From 0 to the numeric value's original size minus 1	132
		Length	From 2 to the numeric value´s original size minus Begin	
5	Add one numeric character	Position	Random	1123
6	Flip sign	-	Positive numbers become negative and vice-versa	-123
7	Replace one random numeric character	Position	Random.	153
8	Add 1	-	-	124
9	Substract 1	-	-	122

5 Time Data Mutations

Table 4: Mutations for Time data types

No	Fault Description	Argument	Configuration	Example
1	Replace by null value	-	-	null
2	Replace by empty dummy time	-	-	0:00:00
3	Replace by random time	Field	Hours field	7:10:08
			Minutes field	2:52:08
			Seconds field	2:10:25
		Value	Uniformly random between the maximum and minimum allowed value for the field	-
4	Reverse field	Field	Hours field	20:10:08
			Minutes field	02:01:08
			Seconds field	02:10:80
5	Add one extra digit	Position	Before hours value	102:10:08
			After hours value	021:10:08
			Before minutes value	02:110:08
			After minutes value	02:101:08
			Before second value	02:10:108
			After second value	02:10:081
6	Duplicate one digit	Position	Before hours value	002:10:08
			After hours value	022:10:08
			Before minutes value	02:110:08
			After minutes value	02:100:08
			Before second value	02:10:008
			After second value	02:10:088
7	Remove one digit	Field	Hours field	0:10:08
			Minutes field	2:01:08
			Seconds field	2:10:00
		Position	Random character	-

6 Date Data Mutations

Table 5: Mutations for Date data types

No	Fault Description	Argument	Configuration	Example
1	Replace by null value	-	-	null
2	Replace by empty dummy date	-	-	0000-00-00
3	Replace by random date	Field	Year field	2120-05-10
			Month field	2020-08-10
			Day field	2020-05-12
		Value	Uniformly random between the maximum and minimum allowed value for the field	-
4	Add one extra digit	Position	Before year value	12020-05-10
			After year value	20201-05-10
			Before month value	2020-105-10
			After month value	2020-051-10
			Before day value	2020-05-110
			After day value	2020-05-101
5	Remove one digit	Field	Year field	2000-05-10
			Month field	2020-5-10
			Day field	2020-05-01
6	Replace one digit with a random digit	Field	Year field	2320-05-10
			Month field	2020-08-10
			Day field	2020-05-14
7	Invert digits in a field	Field	Year field	2002-05-10
			Month field	2020-50-10
			Day field	2020-05-01
8	Swap values among fields	Source	Year field	05-2020-01
			Month field	2020-10-05
			Day field	10-05-2020
		Target	Year field	10-05-2020
			Month field	2020-10-05
			Day field	10-05-2020
9	Replace with the default value	Default date	-	1970-01-01
10	Remove first 2 digits of year field	-	-	0020-05-10
11	Add 100 years	-	-	2120-05-10
12	Subtract 100 years	-	-	1920-05-10
13	Duplicate digit	Field	Year field	20200-05-10
			Month field	2020-055-10
			Day field	2020-05-110

7 Timestamp Data Mutations

Table 6: Mutations for Timestamp data types

No	Fault Description	Argument	Configuration	Example
1	Replace by null timestamp	-	-	null
2	Replace by empty dummy timestamp	-	-	0000-00-00 00:00:00.000000
3	Replace by random timestamp	Position	Year field	2012-05-10 11:11:11.121212
			Month	2020-11-10 11:11:11.121212
			Day field	2020-05-29 11:11:11.121212
			Hour field	2020-05-10 02:11:11.121212
			Minute field	2020-05-10 11:54:11.121212
			Second field	2020-05-10 11:11:23.121212
			Nanosecond field	2020-05-10 11:11:11.565654
		Value	Uniformly random between the maximum and minimum allowed value for the field	-
4	Reverse field	Field	Year field	0202-05-10 19:45:31.143684
			Month	2020-50-10 19:45:31.143684
			Day field	2020-05-01 19:45:31.143684
			Hour field	2020-05-10 91:45:31.143684
			Minute field	2020-05-10 19:54:31.143684
			Second field	2020-05-10 19:45:13.143684
			Nanosecond field	2020-05-10 19:45:31.486341
5	Add one extra character to timestamp	Field	Before year value	12020-05-10 11:11:11.121212
			After year value	20209-05-10 11:11:11.121212
			Before month value	2020-205-10 11:11:11.121212
			After month value	2020-051-10 11:11:11.121212
			Before day value	2020-05-310 11:11:11.121212
			After day value	2020-05-105 11:11:11.121212
			Before hour value	2020-05-10 611:11:11.121212
			After hour value	2020-05-10 114:11:11.121212
			Before minute value	2020-05-10 11:711:11.121212
			After minute value	2020-05-10 11:113:11.121212
			Before second value	2020-05-10 11:11:411.121212
			After second value	2020-05-10 11:11:119.121212
			Before nanosecond value	2020-05-10 11:11:11.9121212
			After nanosecond value	2020-05-10 11:11:11.1212128
6	Duplicate one character on timestamp value	Position	Before year value	22020-05-10 11:11:11.121212
			After year value	20202-05-10 11:11:11.121212
			Before month value	2020-505-10 11:11:11.121212
			After month value	2020-055-10 11:11:11.121212
			Before day value	2020-05-110 11:11:11.121212
			After day value	2020-05-101 11:11:11.121212
			Before hour value	2020-05-10 111:11:11.121212
			After hour value	2020-05-10 111:11:11.121212
			Before minute value	2020-05-10 11:111:11.121212
			After minute value	2020-05-10 11:111:11.121212
			Before second value	2020-05-10 11:11:111.121212
			After second value	2020-05-10 11:11:111.121212
			Before nanosecond value	2020-05-10 11:11:11.2121212
			After nanosecond value	2020-05-10 11:11:11.1212122
7	Add 100 years to the timestamp	-	-	2120-05-10 11:11:11.121212
8	Subtract 100 years from the timestamp	-	-	1920-05-10 11:11:11.121212
9	Remove 2 digits of the year field	-	-	20-05-10 11:11:11.121212121

8 Boolean Data Mutations

Table 7: Mutations for Boolean data types

No	Fault Description	Argument	Configuration	Example
1	Replace by null value	-	-	null
2	Flip value	-	True becomes False, and vice-versa	FALSE
3	Set to True	-	-	TRUE
4	Set to False	-	-	FALSE

9 Decimal Data Mutations

Table 8: Mutations for Decimal data types

No	Fault Description	Argument	Configuration	Example
1	Replace by null value	-	-	null
2	Replace by random same-size decimal values	-	-	456,12
3	Add 1	-	-	124,13
4	Substract 1	-	-	122,13
5	Remove one random numeric character from data	-	-	13,13
6	Duplicate one of the numeric characters	Position	Random from 0 to the numeric value's original size. Repeated digit will be put in front of original digit	1223,13
7	Reverse two consecutive characters	Position	Random from 0 to the numeric value's original size minus 1	132,13
8	Add numeric character	Position	Random from 0 to the numeric value's original size	4123,13
9	Flip sign	-	Positive numbers become negative and vice-versa	-123,13
10	Zero out the integer part	-	-	0,13
11	Zero out the fractional part	-	-	123,0
12	Remove the integer part	-	-	13,0
13	Remove the fractional part	-	-	123,0
14	Remove the decimal separator	-	-	12313,0
15	Add 10^-X	X	Random, from 1 to the limit of the type	123,14
16	Subtract 10^-X	X	Random, from 1 to the limit of the type	123,12
17	Shift decimal separator	Direction	Left	12,313
			Right	1231,3
		Amount	One position, as long as there at is at least 1 digit in both sides	-
18	Add 1 zero next to decimal separator	Position	After decimal separator	123,013
			Before decimal separator	1230,13
19	Swap fractional and integer parts	-	-	13,123
20	Reverse fractional part	-	-	123,31
21	Reverse integer part	-	-	321,13
22	Skipping decimal point near zero	-	-	12313,0

10 Double Data Mutations

Table 9: Mutations for Double data types

No	Fault Description	Argument	Configuration	Example
1	Replace by random same-size double values	-	-	456,12
2	Add 1	-	-	124,13
3	Substract 1	-	-	122,13
4	Remove one random numeric character from data	-	-	13,13
5	Duplicate one of the numeric characters	Position	From 0 to the numeric value's original size. Repeat digit will be put in front of Position	1223,13
6	Reverse two consecutive numeric characters	Begin	From 0 to the numeric value's original size minus 1	132,13
7	Add numeric character	Position	Random position	1223,13
8	Flip sign	-	Positive numbers become negative and vice-versa	-123,13
9	Zero out the integer part	-	-	0,13
10	Zero out the fractional part	-	-	123,0
11	Remove the integer part	-	-	13,0
12	Remove the fractional part	-	-	123,0
13	Remove the decimal separator	-	-	12313,0
14	Add 10^-X	X	Random, from 1 to the limit of the type	123,14
15	Subtract 10^-X	X	Random, from 1 to the limit of the type	123,12
16	Shift decimal separator	Direction	Left	12,313
			Right	1231,3
		Amount	One position, as long as there at is at least 1 digit in both sides	-
17	Add a zero next to decimal separator	Position	After decimal separator	123,013
			Before decimal separator	1230,13
18	Swap fractional and integer parts	-	-	13,123
19	Reverse fractional part	-	-	321,13
20	Reverse integer part	-	-	123,31
21	Skipping decimal point near zero	-	-	12313

11 Binary Data Mutations

Table 10: Mutations for Binary data types

No	Fault Description	Argument	Configuration	Example
1	Replace by null value	-	-	null
2	Replace by empty	-	-	[]
3	Add byte	Position	Prefix	[1, 10, 4, 5, 149]
			Middle (Random)	[10, 4, 5, 1, 149]
			Suffix	[10, 4, 5, 149, 1]
		Value	Random from 0 to 255	-
4	Remove bytes	Position	Prefix	[5, 149]
			Middle (Random)	[10, 149]
			Suffix	[10, 4]
		Quantity	Random from 1 to size of Binary - Position	-
5	Duplicate byte	Position	Prefix	[10, 10, 4, 5, 149]
			Middle (Random)	[10, 4, 5, 5, 149]
			Suffix	[10, 4, 5, 149, 149]
6	Replace byte	Position	Prefix	[6, 4, 5, 149]
			Middle (Random)	[10, 4, 6, 149]
			Suffix	[10, 4, 5, 6]
		Value	Random from 0 to 255	-
7	Drop X most significant bits	X	From 1 to 8 bits that make a byte	[0, 0, 0, 16]
8	Drop X least significant bits	X	From 1 to 8 bits that make a byte	[2, 4, 5, 5]
9	Flip sign bit of a byte	Position	Prefix	[138, 4, 5, 149]
			Middle (Random)	[10, 132, 5, 149]
			Suffix	[10, 4, 5, 21]

12 Object Data Mutations

Table 11: Mutations for Object data types

No	Fault Description	Argument	Configuration	Example
1	Replace by null object	-	-	null
2	Replace by empty object	-	-	-
3	Replace by object of different type	-	-	-

13 Reference Data Mutations

Table 12: Mutations for Reference data types

No	Fault Description	Argument	Configuration	Example
1	Replace by null	-	-	null
2	Replace by reference to object of different type	-	-	-

References

[1] Thomas Mahler and Armin Waibel. Jdbc types, December 2012. https://db.apache.org/ojb/docu/guides/jdbc-types.html#mapping-table.