CSCI 621
GROUP 3 - H2 Implementation Assignment 2

Abhishek Shah, Harsh Bhansali, Miloni Sangani, Rasika Sasturkar

INTRODUCTION

- We planned to implement a custom data type: Password to the H2 database as Passwords are very commonly stored in databases.
- First,the format of the Password entered is checked and only if it's valid,it is stored in the database, else an error message is displayed.
- To achieve this,we first decided to add a new data type to the Value.java file in the source code. In H2, custom data types are java objects so we create a Password class which implements the Serializable interface.The password object can thus be serialized to disk and deserialized back.
- H2 allows users to create their own data types by implementing the minimal required CustomDataTypesHandler API. To use this feature, the CustomDataTypeHandler.java file should implement all the abstract methods defined in the CustomDataTypesHandler interface.
- Error handling is done by adding a new unique constant error code to the ErrorCode java file. This error code is displayed when the password entered is invalid.
- Also we need to create a specialized Value object for password which extends the generic Value object. This specialized object is responsible for instantiating a password entity object and storing it.
- We tried to make these modifications in the available source code but the api's we were trying to change are no longer part of the current H2 database version. We couldn't figure out a way to provide a working example of the code and the final implementation.
- So alternatively, we have finally implemented new String and Numeric functions that are not present in H2. We added String functions like: String length(Strlen) , String reverse(Strreverse) and numeric functions like Cube root(Cbrt) and Square (Sqr).

- We know that these are not Storage or indexing features but we think that these features will enhance the database.

ENHANCED FEATURES DESCRIPTION

We added two features in String functions and two in Numeric functions.

1. String functions
    a. **STRLEN** (String length)
    This function computes the length of the string. That is, it takes a string as an argument and returns its length.

        - **List of files modified:**
            1. *StringFunction1.java* (h2database/h2/src/main/org/h2/expression/function/StringFunction1.java)

            2. *Parser.java* (h2database/h2/src/main/org/h2/command/Parser.java)

    b. **STRREVERSE** (String Reverse)
    This function reverses the string. That is, it takes a string as an argument and returns the reversed version of it.

        - **List of files modified:**
            1. *StringFunction1.java* (h2database/h2/src/main/org/h2/expression/function/StringFunction1.java)

            2. *Parser.java* (h2database/h2/src/main/org/h2/command/Parser.java)

2. Numeric functions
    a. **CBRT** (CubeRoot)
    This function computes and returns the cube root when given a number as an argument.

        - List of files modified:
            1. *MathFunction1.java* (h2database/h2/src/main/org/h2/expression/function/MathFunction1.java)

            2. *Parser.java* (h2database/h2/src/main/org/h2/command/Parser.java)

    b. **SQR** (Square)
    This function computes and returns the square of a number when given a number as an argument.

        - List of files modified:
            1. *MathFunction1.java* (h2database/h2/src/main/org/h2/expression/function/MathFunction1.java)

            2. *Parser.java* (h2database/h2/src/main/org/h2/command/Parser.java)

ENHANCED FEATURES IMPLEMENTATION

This is how we implemented the features.

- **STRLEN & STRREVERSE**

```
95
96      /**
97       * STRLEN
98       */
99      public static final int STRLEN = QUOTE_IDENT + 1;
100
101     /**
102      * STRREVERSE
103      */
104     public static final int STRREVERSE = STRLEN + 1;
105
```

Fig 1: STRLEN, STRREVERSE code addition in StringFunction1.java

```
170         case STRLEN:
171             v = ValueInteger.get(v.getString().length());
172             break;
173         case STRREVERSE:
174             StringBuilder input = new StringBuilder(v.getString());
175             input.reverse();
176             v = ValueVarchar.get(input.toString());
177             break;
```

Fig 2: STRLEN, STRREVERSE code for case addition in StringFunction1.java

```
4254    case "STRLEN":
4255        return new StringFunction1(readSingleArgument(), StringFunction1.STRLEN);
4256    case "STRREVERSE":
4257        return new StringFunction1(readSingleArgument(), StringFunction1.STRREVERSE);
```

Fig 3: STRLEN, STRREVERSE code for case addition in Parser.java

- **CBRT & SQR**

```
155         case "SQR":
156             return new MathFunction1(readSingleArgument(), MathFunction1.SQR);
157         case "SQRT":
158             return new MathFunction1(readSingleArgument(), MathFunction1.SQRT);
159         case "CBRT":
160             return new MathFunction1(readSingleArgument(), MathFunction1.CBRT);
```

Fig 4: CBRT, SQR code for case addition in Parser.java

```
92              case SQR:
93                  d = Math.pow(d, 2);
94                  break;
95              case SQRT:
96                  d = Math.sqrt(d);
97                  break;
98              case CBRT:
99                  d = Math.cbrt(d);
00                  break;
```

Fig 5: STRLEN, STRREVERSE code for case addition in MathFunction1.java

ENHANCED FEATURES IN ACTION

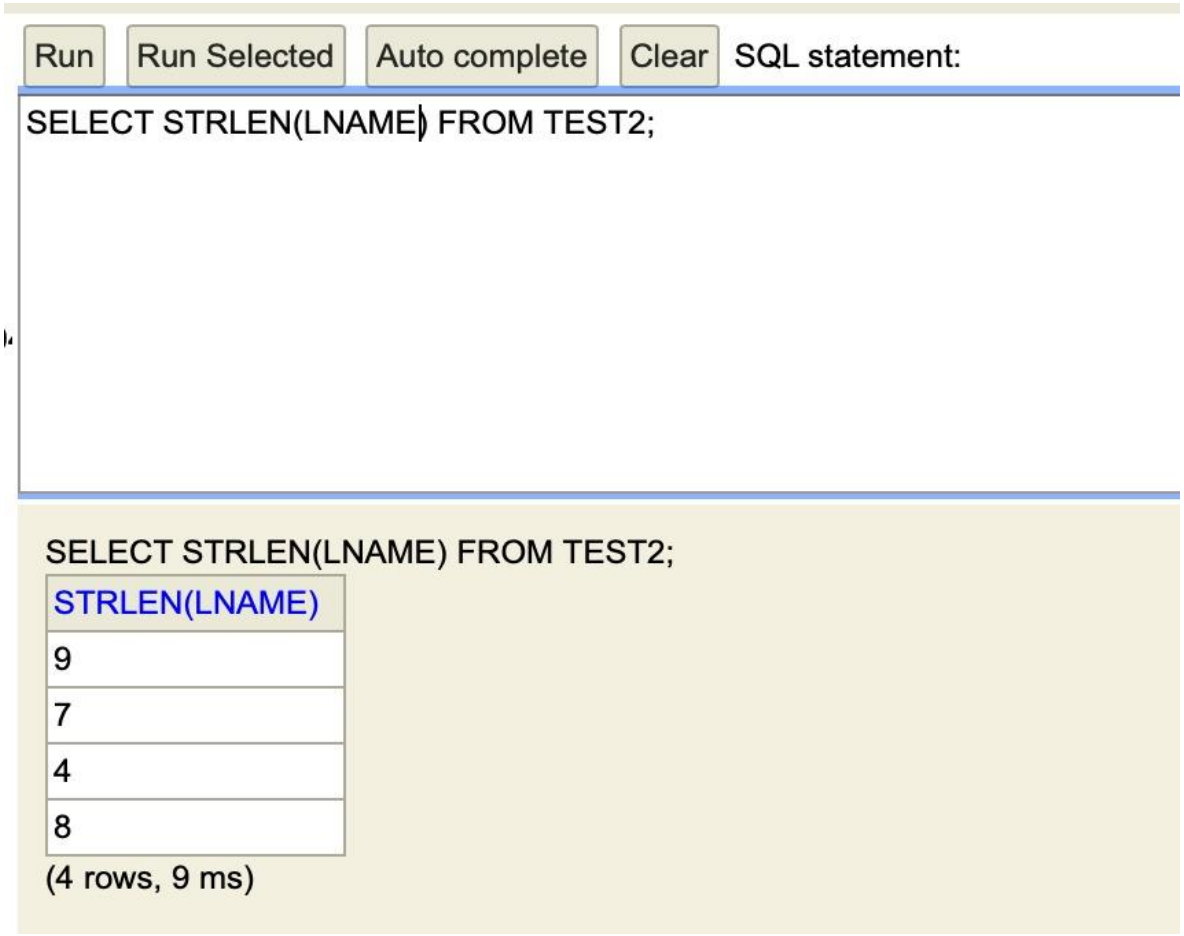
Consider the following database.

SELECT * from TEST2;

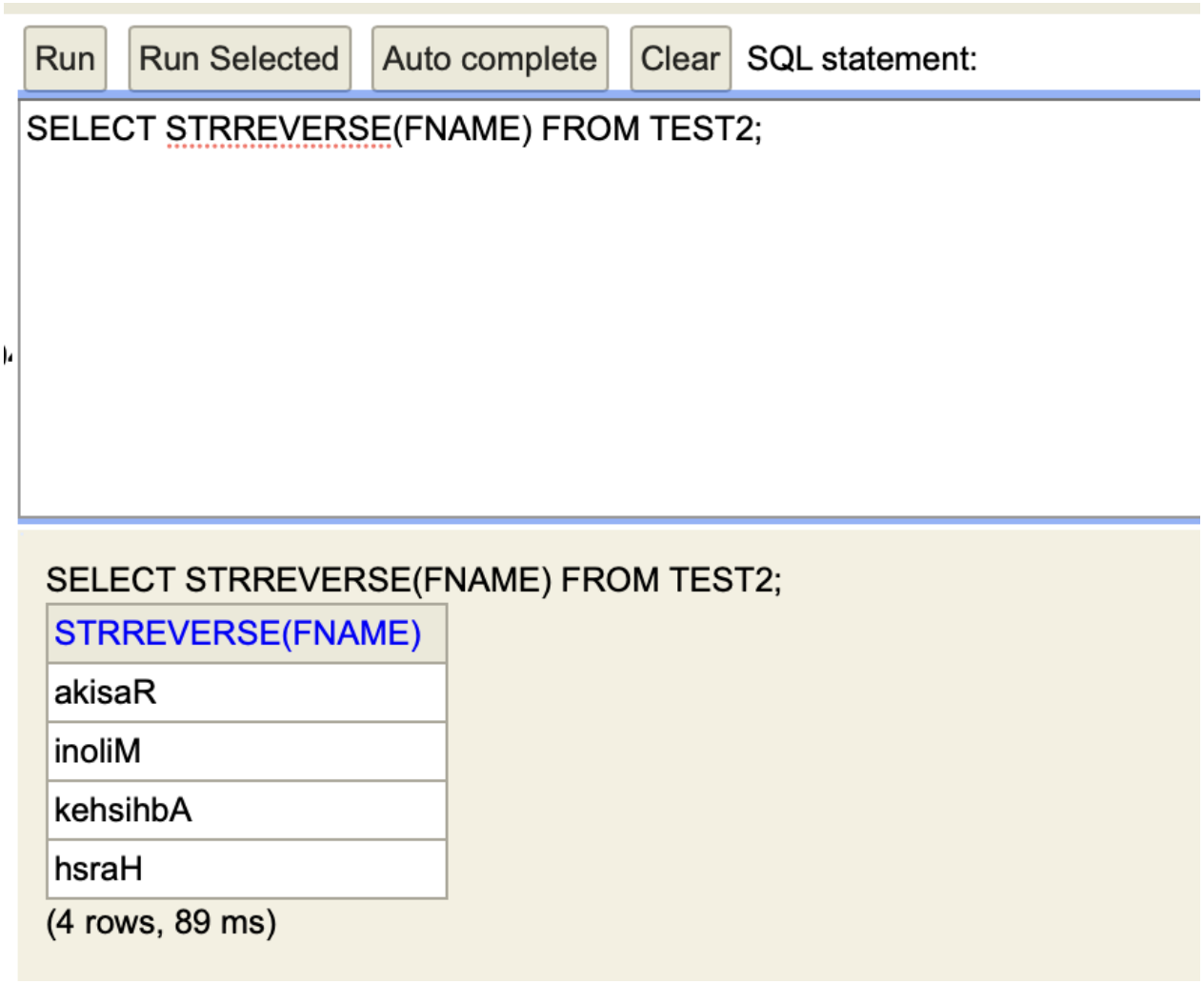
ID	FNAME	LNAME
1	Rasika	Sasturkar
2	Miloni	Sangani
3	Abhishek	Shah
4	Harsh	Bhansali

(4 rows, 7 ms)

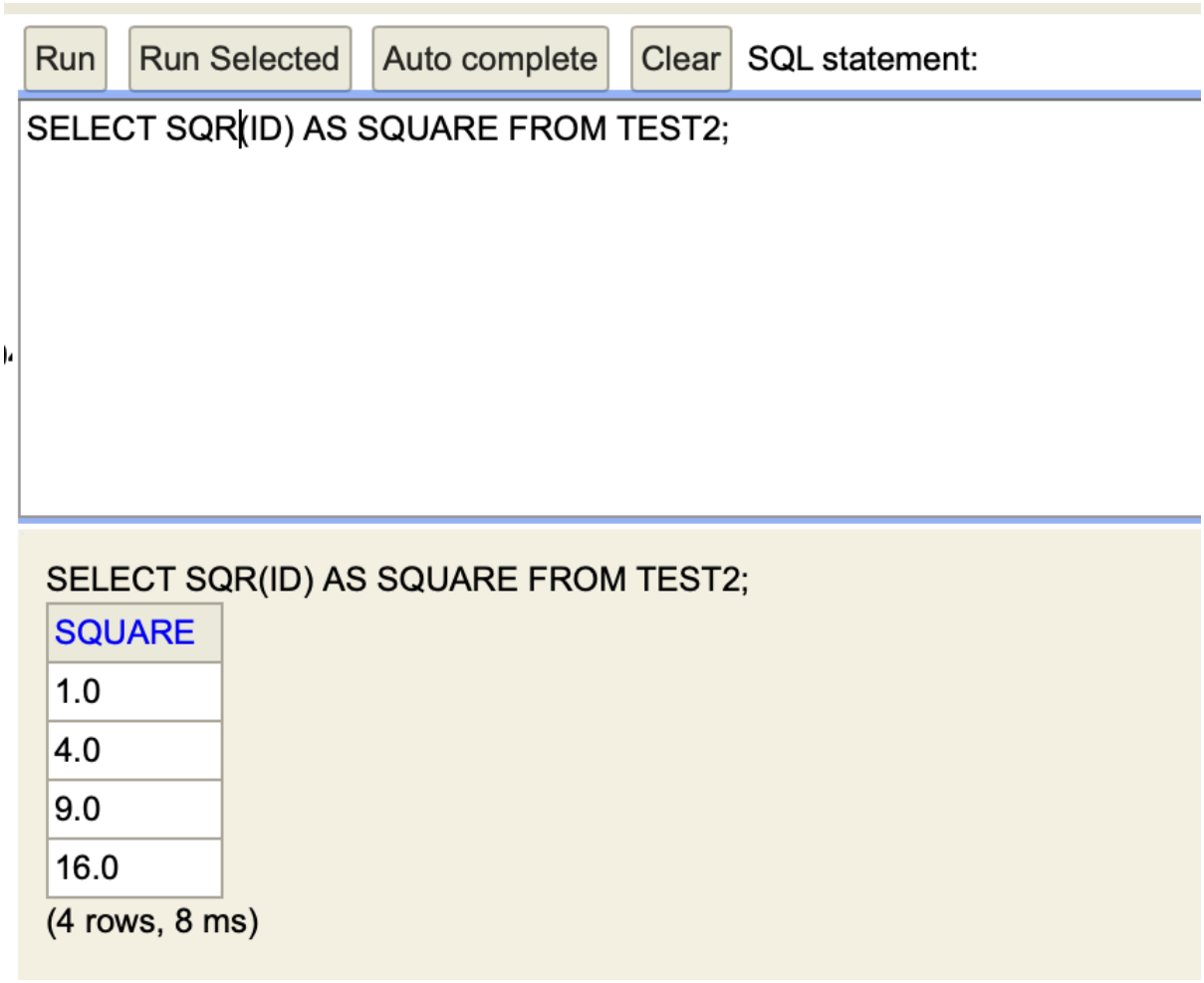
This is how **STRLEN** computes and returns the length of the string.



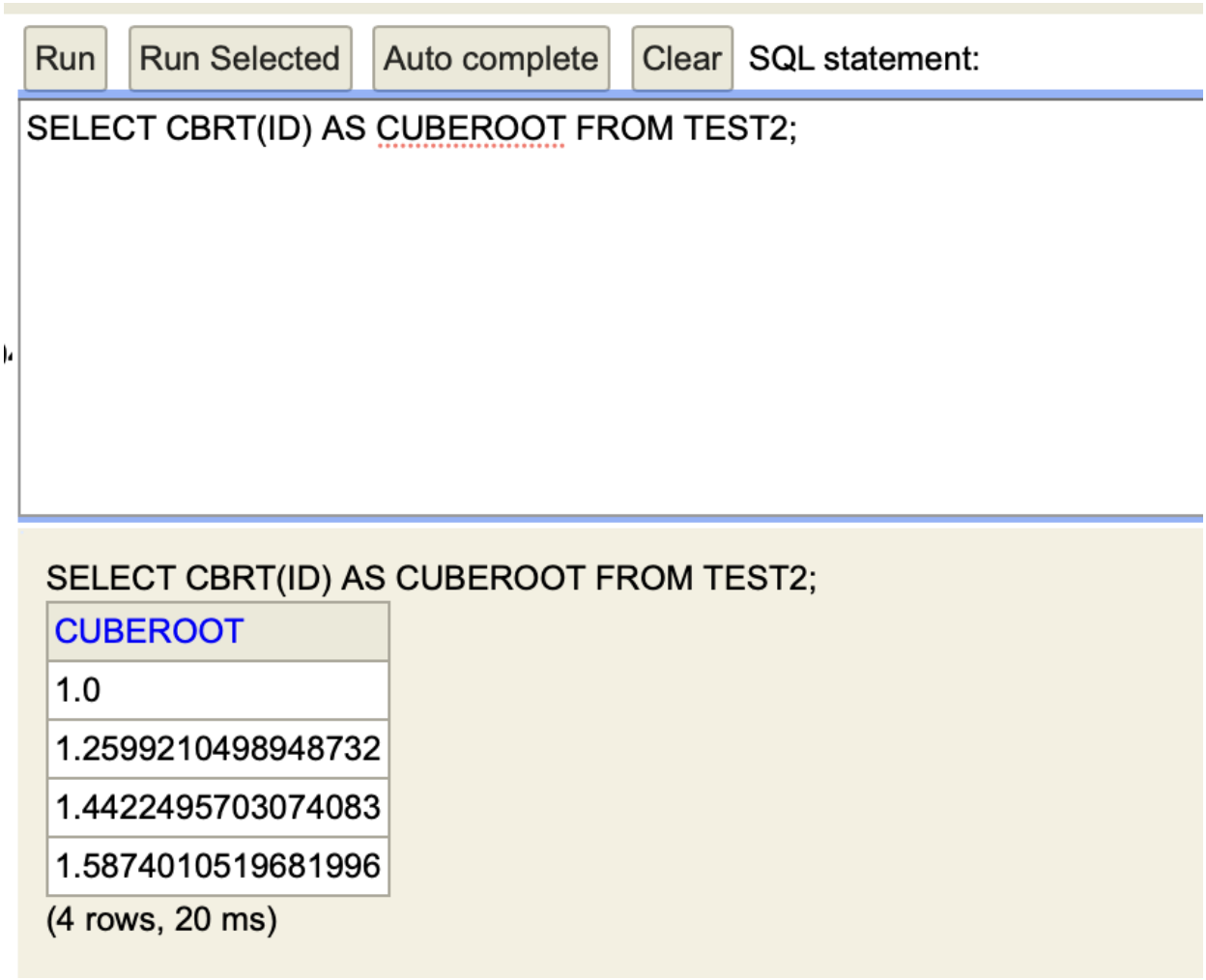
This is how **STRREVERSE** reverses the string.



This is how **SQR** returns the square of a number.

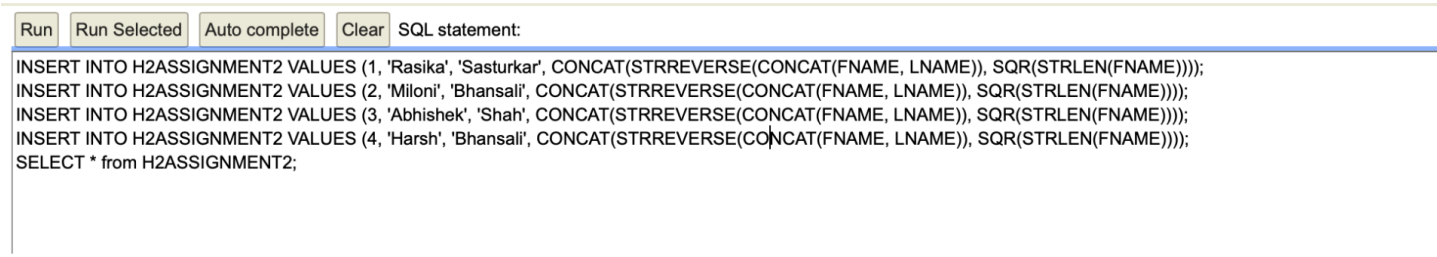


This is how **CBRT** computes and returns the cube root of a number.



Let’s say we were asked to suggest the password which is a combination of alphanumeric characters when given a person’s first name and last name. One example to compute passwords, is to take the exact reverse of their names after concatenating the user's first name and last name followed by the square of the length of their full name.

This is how we can implement it by using the STRREVERSE, STRLEN and SQR functions that we defined in the H2 database.



This is how the password for each user will look like.

SELECT * from H2ASSIGNMENT2;

ID	FNAME	LNAME	PASSWORD
1	Rasika	Sasturkar	rakrutsaSakisaR36.0
2	Miloni	Sangani	inagnaSinoliM36.0
3	Abhishek	Shah	hahSkehsihbA64.0
4	Harsh	Bhansali	ilasnahBhsraH25.0

(4 rows, 7 ms)

WORKLOAD DISTRIBUTION

Everyone

Discussed and studied the source code of H2. Went through all the files in the source code to check which all to edit. Discussed and finalized addition of a custom data type, password. Everyone also edited the report by completing and writing their implementation part.

Abhishek Shah

For the initial submission, Abhishek studied and handled the Value.java file to add a new data type in the source code. Created a Password class which was a java object which implemented a Serializable interface. This object was able to serialize and deserialize to disk.

For the final submission, studied h2 source code for finding all the files to edit for implementing the STRREVERSE function. Wrote code for the same in the StringFunction1.java and Parser.java files.

Rasika Sasturkar

For the initial submission, Rasika studied and handled the ErrorCode.java file to work on error handling. She added a new unique constant error code to the ErrorCode.java file for displaying custom messages when invalid password is entered.

For the final submission, studied h2 source code for finding all the files to edit for implementing the STRLEN function. Wrote code for the same in the StringFunction1.java and Parser.java files.

Miloni Sangani

For the initial submission, Miloni studied and handled CustomDataTypesHandler.java file for creating and handling our own data types. She implemented all the abstract methods by defining it in the CustomDataTypesHandler interface.

For the final submission, studied h2 source code for finding all the files to edit for implementing SQR function. Wrote code for the same in the MathFunction.java and Parser.java files.

Harsh Bhansali

For the initial submission, Harsh studied and handled a specialized value object for the password class. This object extended the generic Value object and was responsible for instantiating a password entity object and storing it.

For the final submission, studied h2 source code for finding all the files to edit for implementing CBRT function. Wrote code for the same in the MathFunction.java and Parser.java files.