H2 Database Implementation : Assignment 2

Implementing House Data Type

Kavya Kotian

Siddarth Sargunaraj

Neelabh Kumar

Prashanth Kumar Purshotam

October 25, 2019

Introduction

This Assignment aims at implementing a custom Datatype called the *House* datatype. Our *House* datatype can have any values from the following list: *Condominium, Apartment, Townhome, Bungalow, Cottage, Cabin, Chalet, Barndominium, Mansion, Yurt, Castle, Palace, Chateau, Villa, Manor, Fort, Cave*.

Implementation

To implement the *HOUSE* datatype, we modify a few existing classes and also add out own classes. The modified classes are: *ErrorCode.java, SysProperties.java, TypeInfo.java, Value.java*. The new classes are: *CustomHouseDataType.java, House.java, valueHouse.java*. The detailed explanation of these classes are given below.

CustomHouseDataType

CustomHouseDataType implements the *CustomDataTypesHandler* interface from the package *org.h2.api*. The constructor creates an object of *DatatType* class called *houseDataType* and sets Datatype *name* to *House*, sets *type* to the ID denoted to House data type in the Value class and sets *sqlType* to JAVA_OBJECT from java.sql.types which is used to identify generic SQL types.

- publc DataType getDataTypeByName(String name)
  Checks if *name* matched our custom datatype and returns the Datatype *HOUSE*, else returns null. Given the String parameter *name* we convert it to uppercase to make it case insensitive and check if it matches out *houseDataTypeName*(i.e HOUSE).

- public DataType getDataTypeById(int type)
  Checks if *type* matches the ID denoted by *Value.HOUSE* for our custom datatype and returns the Datatype *HOUSE*, else returns null.

- public int getDataTypeOrder(int type)
  Returns the order denoted for the HOUSE class from the value class if the parameter *type* matches the order value for *HOUSE* class in the *Value* class.

- public Value convert(Value source, int targetType)
  Convert the provided source value into value of given target data type. You can convert a datatype to a House datatype from Object datatype, String datatype and Byte Datatype.

- public String getDataTypeClassName(int type)
  Given a *type* object we return its class name. In this function we check if our object class is *HOUSE* else throw the *Unknown Datatype Exception*

- public int getTypeIdFromClass(Class<?> cls)
  Returns *value id* for *HOUSE* if *cls* is *HOUSE* class.

- public TypeInfo getTypeInfoById(int type, long precision, int scale, ExtTypeInfo extTypeInfo)
  Return the *Type Info* ID if the *type* matches the ID for *HOUSE* in *Value* class else throws an *Unknown Datatype Exception*.

- public Value getValue(int type, Object data, DataHandler dataHandler)
  If *type* matches *HOUSE type* and *data* is an instance of *HOUSE* we return an object of *HOUSE* datatype.

- public Object getObject(Value value, Class<?> cls)
  Converts *Value* object of value class to the specified class, which in our case we convert to a *HOUSE* datatype. If the *cls* is a HOUSE datatype and value type is that of *HOUSE* we just create an object of *value* and return it. If not, we convert *value* to *HOUSE* datatype.

- public boolean supportsAdd(int type)
  Checks if *type* supports add operation and returns a boolean value.

- public int getAddProofType(int type)
  Checks if *type* has the datatype which can sustain multiple adds without overflow which in our case we check if the datatype is *HOUSE*.

House

The *House* class is used for serialization and to validate the string passed as instance of *House* class. It consists of an *Arraylist* of valid values : *Condominium, Apartment, Townhome, Bungalow, Cottage, Cabin, Chalet, Barndominium, Mansion, Yurt, Castle, Palace, Chateau, Villa, Manor, Fort, Cave* is saved as the variable *houseList*.

- public boolean equals(Object o)
  Check if the Object passed is equal to the current object of House class.

- public int hashCode()
  Generates a unique id for the current object which is the index of the Instance variable in the ArrayList *houseList*.

- public String toString()
  Returns the String format of the given object.

- public void checkValidity(String house)
  This method checks if the incoming parameter string *house* belongs to the ArrayList *houseList*. If it does not belong to the *houseList* an Exception is thrown by calling the DbException.get(ErrorCode.INVALID_HOUSE_ERROR) where INVALID_HOUSE_ERROR is a custom error code for the *House Datatype*.

Value

Value is a base class in the h2 database. It provides comparison and conversion methods for all value classes. Since it is a base class, we have only modified it to recognize our *HOUSE* datatype.

- We first initialize the value type for *HOUSE*:
  public static final int HOUSE = 40;

- Since we have added a new datatype, we increment the total number of value types by one:
  public static final int TYPE_COUNT = HOUSE + 1;

- We then add a case to get the order of the *HOUSE* type:
  case HOUSE:
  return 53_000;

ValueHouse

valueHouse is an extension of the *Value* class from the package *org.h2.value.* The constructor creates an object of *House* class called *house*

- public TypeInfo getType()
  Returns the information on the parameters used by the *HOUSE* datatype
- public StringBuilder getSQL(StringBuilder builder)
  Append the SQL expression for the value *HOUSE* to the string builder
- public int getValueType()
  Returns the value of the *HOUSE* datatype
- pubic String getString()
  Get the value of the *HOUSE* datatype as a string
- public Object getObject()
  Get the value of the *HOUSE* datatype as an object
- public int hashCode()
  Get the hash code of the *HOUSE* datatype.
- public boolen equal(Object other)
  Check whether the two values has the same hash code
- public int compareTypeSafe(Value o, CompareMode mode)
  Compare the value against another value of the same *HOUSE* datatype
- public Value converTo(int targetType, Mode mode, Object column, ExtTypeInfo extTypeInfo)
  Convert a value to the *HOUSE* datatype

TypeInfo

TypeInfo provides information on the parameters used by datatypes available in h2. Since it is already available in the h2 database we only modify it add information on our new *HOUSE* datatype.

- We first define *TYPE_HOUSE* as an object of *TypeInfo*
  public static final TypeInfo TYPE_HOUSE;
- We then initialize the typeinfo list with *TYPE_HOUSE*
  infos[Value.HOUSE]=TYPE_HOUSE=new TypeInfo(Value.HOUSE,Integer.MAX_VALUE,0,Integer.MAX_VALUE,null);
- We then add a case to get the datatype for the value *HOUSE*
  case Value.HOUSE:
  return TYPE_INFOS_BY_VALUE_TYPE[type];

ErrorCode

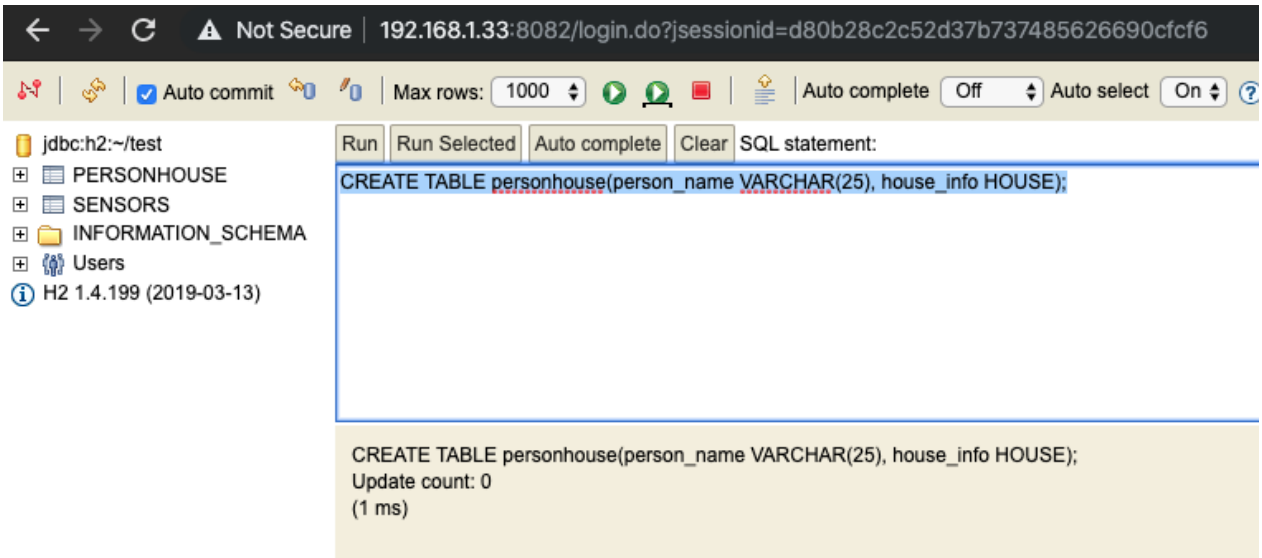
ErrorCode defines the codes used for SQL exceptions. Since it is already available in the h2 database we only modify it add the error code for our new *HOUSE* datatype.
public static final int INVALID_HOUSE_ERROR = 90147;

SysProperties

SysProperties contains settings used by the h2 database in order to execute and load classes on a system. It is already available in the h2 database implementation and most of the properties are based on settings used by the machine. However there are some custom properties. To ensure that our *HOUSE* datatype is recognized we add a custom datatype handler property
public static final String CUSTOM_DATA_TYPES_HANDLER = Utils.getProperty("h2.customDataTypesHandler","org.h2.api.CustomHouseDataType");

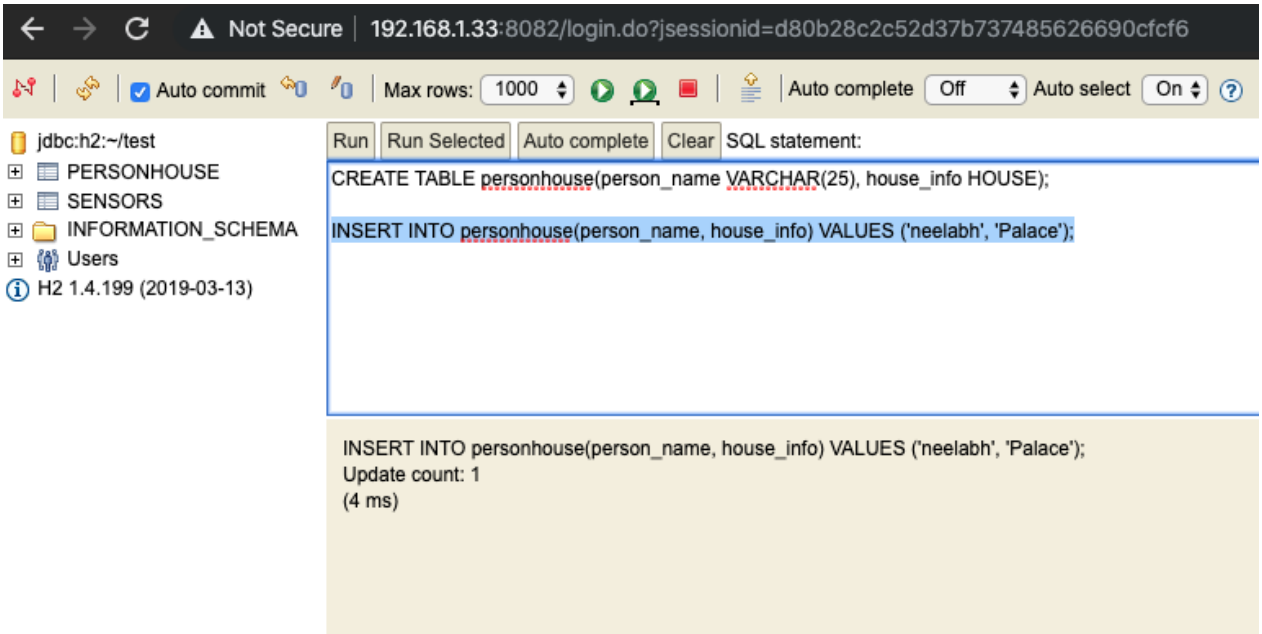
Queries with House Datatype on H2

Create table with house House Datatype



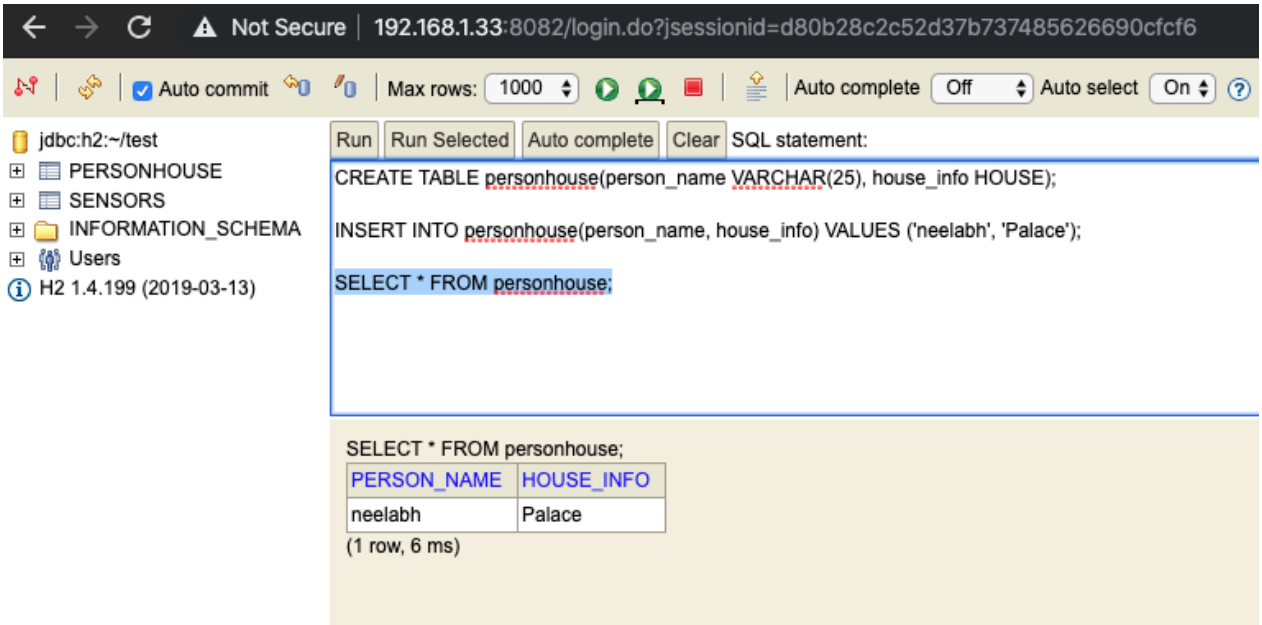
Creating a table

Inserting 1 record into the table



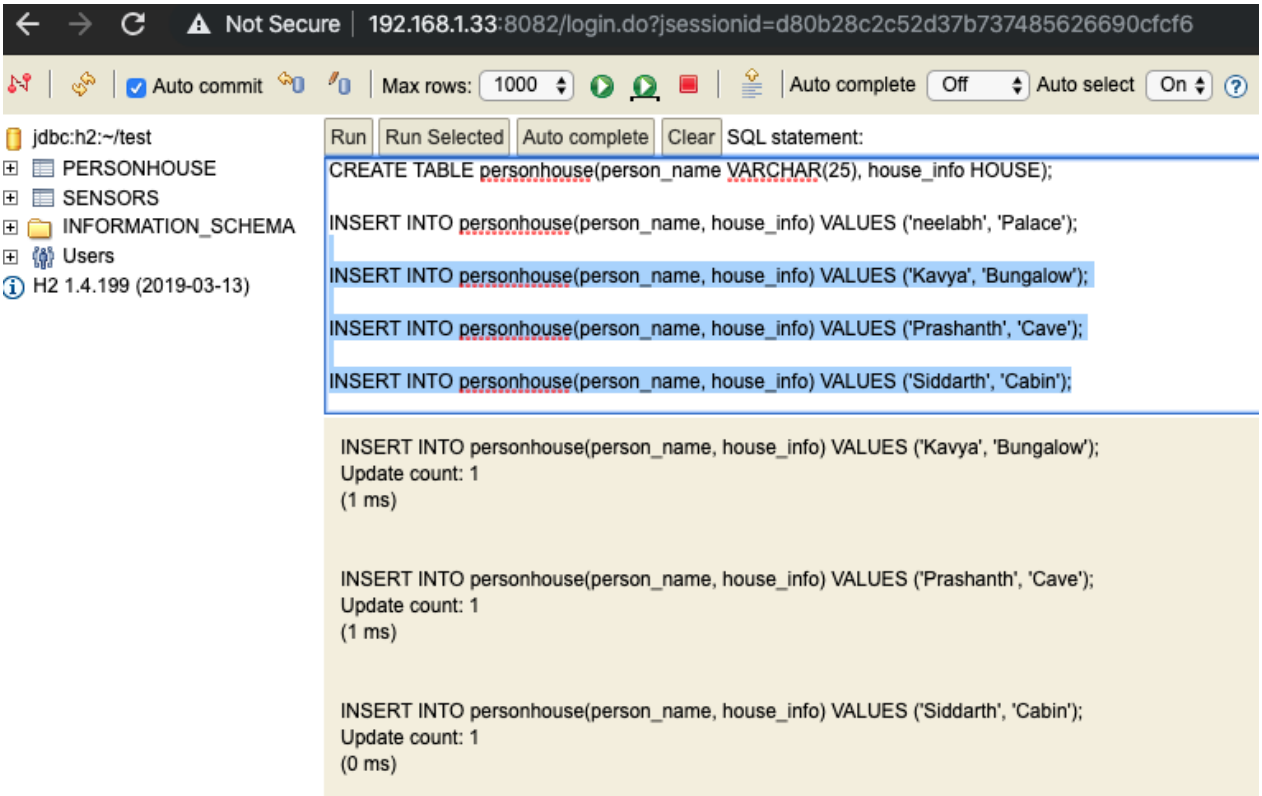
Inserting a record

Show table after inserting 1 record



select * from table

Inserting multiple record into the table

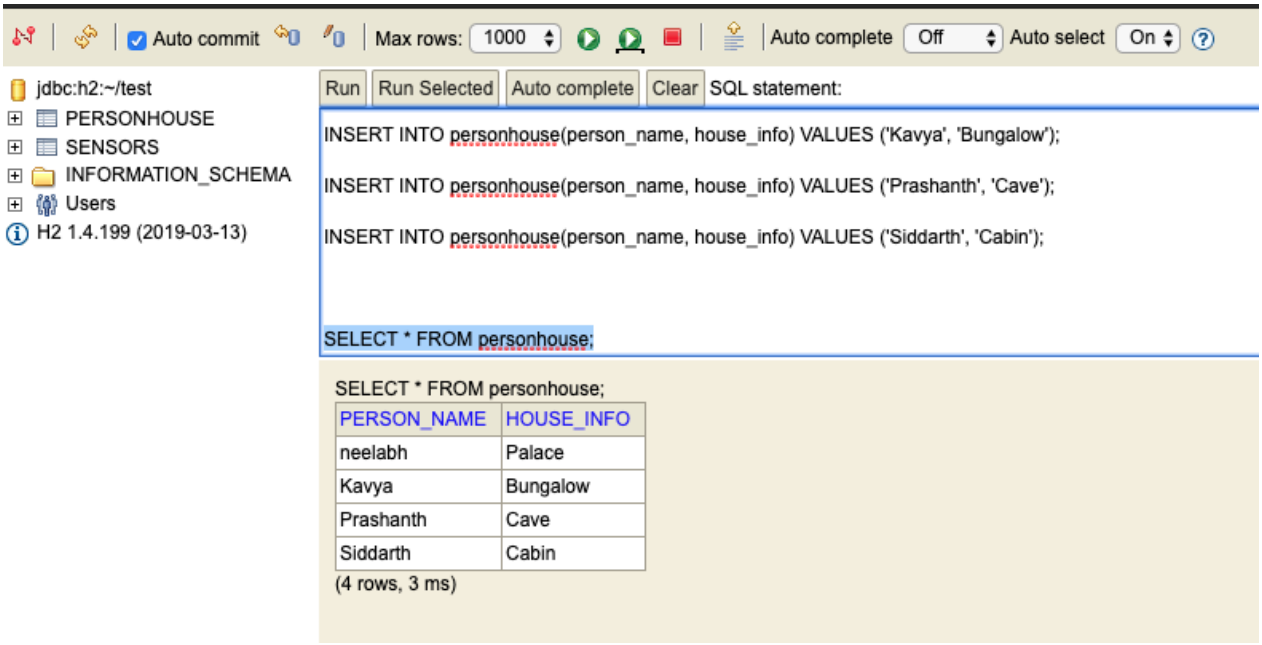


Inserting multiple records

```
  501 35306 55650   0  4:09PM ttys000    0:00.00 grep --color=auto --exclude-dir=.bzr --exclude-dir=CVS --exclude-dir=.git --exclude-dir=.hg --exclude-dir=.svn h2
 ⊙ neelabhkumar@neelabhs-MBP  ~/Documents/h2db/h2/bin  ./h2.sh
1340
coming inside
coming inside string
[Condominium, Apartment, Townhome, Bungalow, Cottage, Cabin, Chalet, Barndominium, Mansion, Yurt, Castle, Palace, Chateau, Villa, Manor, Fort, Cave]
Palace
a new object created
1340
coming inside
coming inside string
[Condominium, Apartment, Townhome, Bungalow, Cottage, Cabin, Chalet, Barndominium, Mansion, Yurt, Castle, Palace, Chateau, Villa, Manor, Fort, Cave]
Bungalow
a new object created
1340
coming inside
coming inside string
[Condominium, Apartment, Townhome, Bungalow, Cottage, Cabin, Chalet, Barndominium, Mansion, Yurt, Castle, Palace, Chateau, Villa, Manor, Fort, Cave]
Cave
a new object created
1340
coming inside
coming inside string
[Condominium, Apartment, Townhome, Bungalow, Cottage, Cabin, Chalet, Barndominium, Mansion, Yurt, Castle, Palace, Chateau, Villa, Manor, Fort, Cave]
Cabin
a new object created
```

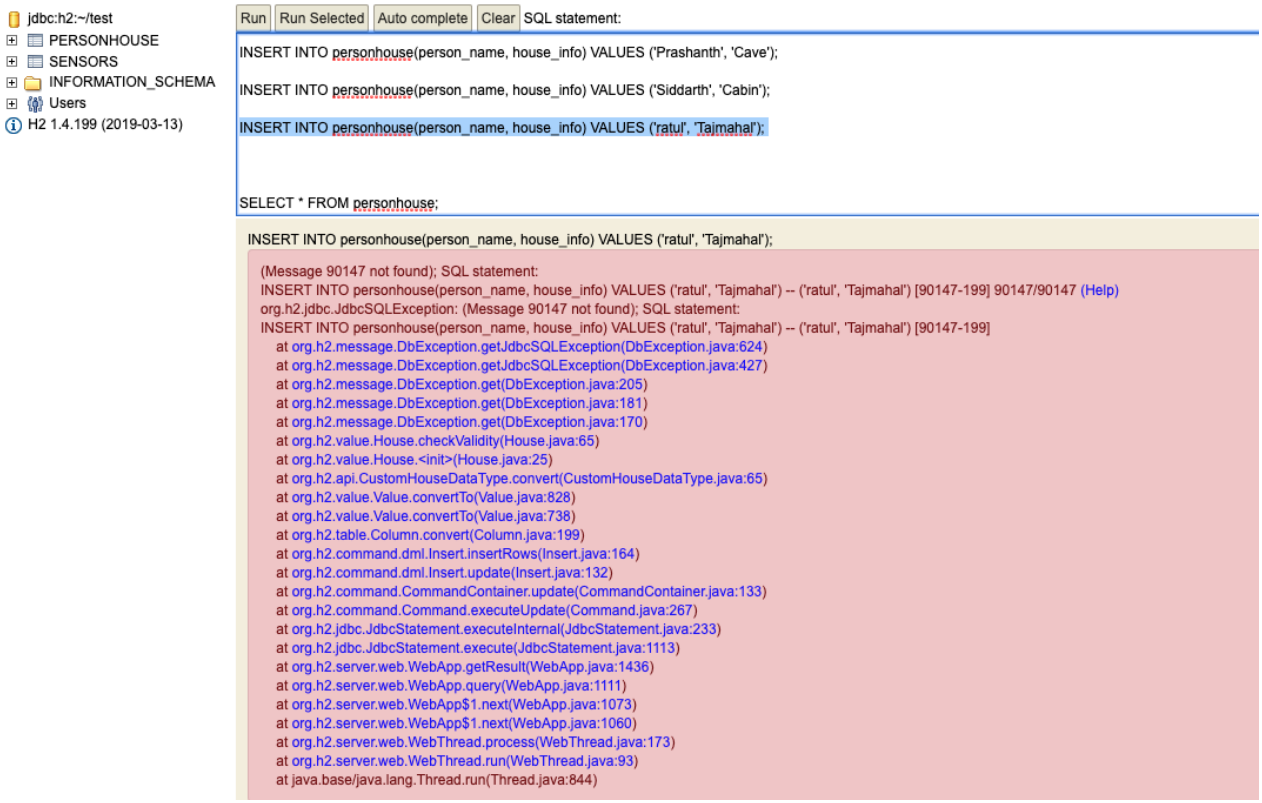
stack Output for the insertion

Show table after inserting multiple record

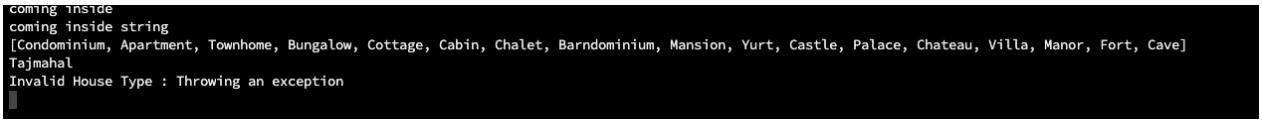


select * from table

Exception thrown after entering a record with invalid House datatype

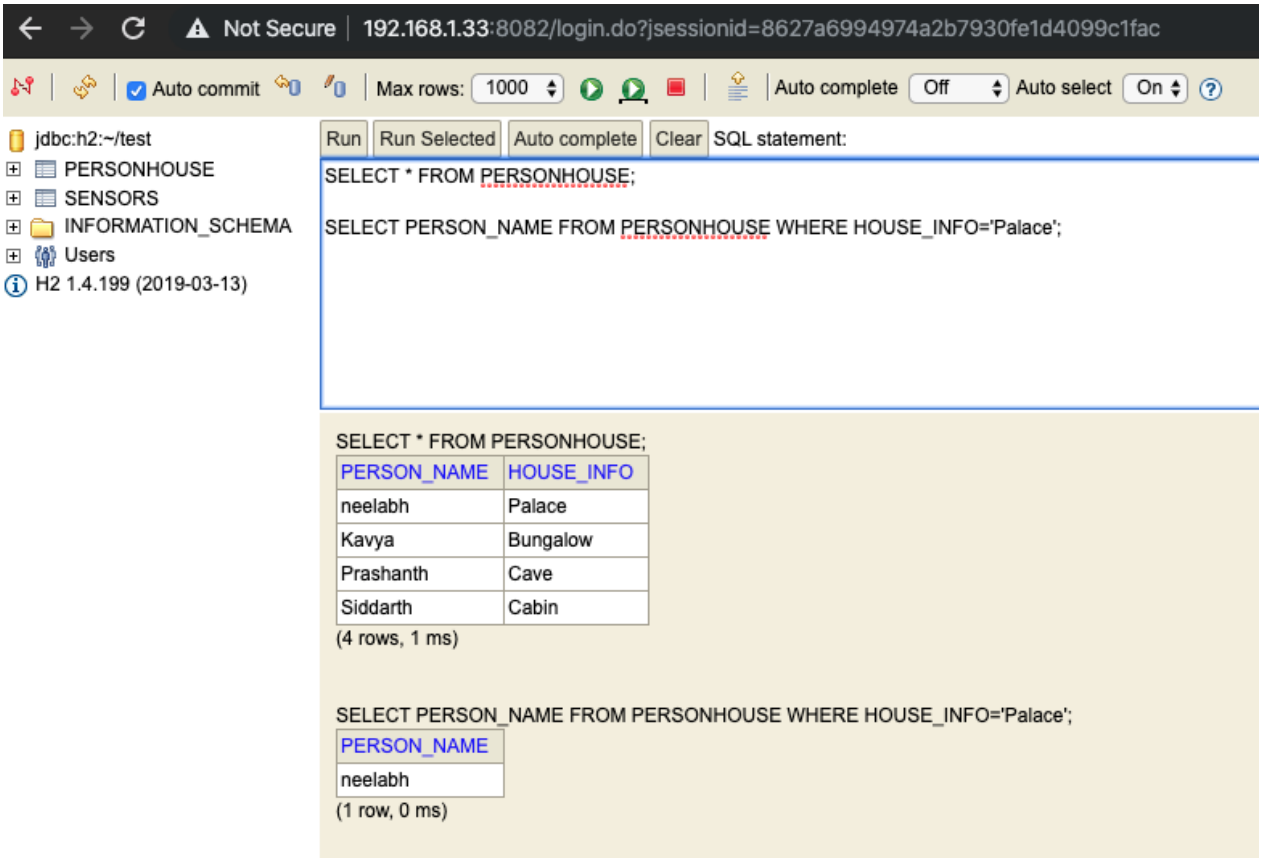


Invalid insertion of HOUSE as TajMahal



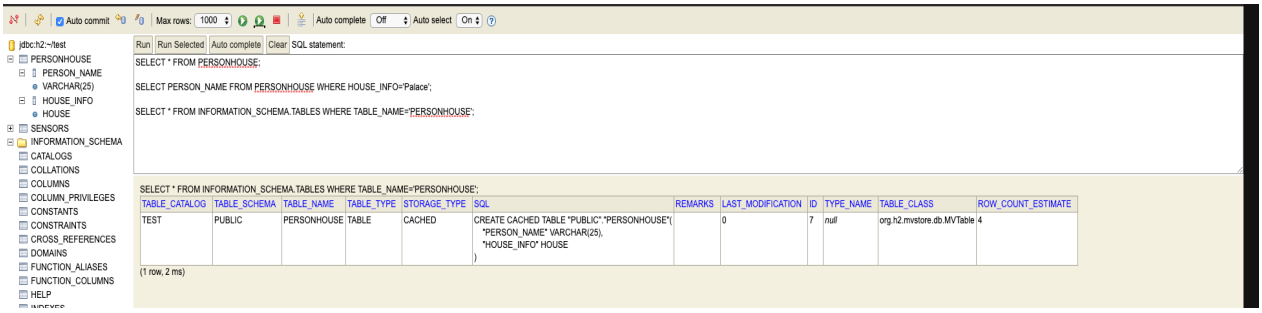
stack output on wrong insertion

Display records that matches the *equality* condition



Equality search

Information Schema



Information Schema showing the type HOUSE

PERSONHOUSE table showing the type HOUSE