Software evolution, Series 2

Michael Peters, 12046515 Ewoud Bouman, 10002578

December 18, 2018

1 Introduction

Code duplication is a common occurrence in software development. The presence of clones can lower the maintainability and adaptability of software projects.

We have developed a tool that analyses java projects for 3 different types of clones. The results are presented in a visual environment giving the programmer an overview of the state of their project.

2 Tool requirements

In order to make this tool useful for our target user group (software developers), we came up with key requirements that must be fulfilled in order to make this tool helpful.

Fracchia et al [SFM99] writes about a bottom-up and top-down approach for creating a mental model of a code base. A bottom-up approach is an approach to create an understanding of every clone found. Besides that, a top-down approach could also be useful for getting an overview of all clones and to know their relations and impact in the complete code base. After getting insight into such a broad overview, a developer could pick a problem and start from the bottom in order to research the problem.

The requirements we want to satisfy are:

1. The visualization should give a very detailed view of clones, from which a user can immediately start researching in a bottom-up manner.
2. The visualization should give an overview of all clones in a way that the user can pick a problem of his choosing to investigate further to create a mental model.
3. The tool should be easily accessible.
4. The tool should be easily updated with for example nightly runs of the clone detection algorithm.
5. A clear distinction should be made in the type of clones that are displayed.

3 Clone detection

This section describes our clone finding algorithm and goes into detail about the various types of clones we can find.

3.1 Clone types

Our implementation supports the detection of 3 clone types [Bel+07].

- Type 1 clones: segments of code that are identical excluding differences in indentation and comments.
- Type 2 clones: segments of code that are structurally identical excluding differences in indentation, comments, identifiers, and literals.
- Type 3 clones: segments of code that are similar but contain differences in statements and/or expressions.

3.2 Implementation

We have implemented an Abstract Syntax Trees clone detection algorithm 1 based on the algorithm proposed by Baxter et al [Bax+98]. This algorithm works by matching all available sub-trees above a certain threshold. This process is optimized by first bucketing all sub-trees with the same hash value. This way a quick preselection is made, which makes the matching process more focused and efficient.

The only change we made is that we sorted all the buckets of the subtrees with the same hash. This makes sure that while iterating, smaller sub-trees are found first and deleted by larger clones found later on.

Algorithm 1 Clone detection algorithm

```
 1: clones = Ø
 2: buckets = Ø
 3: for all subtrees i in AST do
 4:     if mass(i) > MassThreshold then
 5:         buckets[hash(i)] += i
 6:     end if
 7: end for
 8: sortAscending(buckets)
 9: for all subtree i and j in the same bucket do
10:     if CompareTree(i,j) >= SimilarityThreshold then
11:         for all subtree s of i do
12:             if IsMember(clones,s) then
13:                 RemoveClonePair(Clones,s)
14:             end if
15:         end for
16:         for all subtree s of j do
17:             if IsMember(clones,s) then
18:                 RemoveClonePair(Clones,s)
19:             end if
20:         end for
21:         AddClonePAir(clones,i,j)
22:     end if
23: end for
```

Type 1 clones:

For the detection of type 1 clones we used a *MassThreshold* of 40 with a SimilarityThreshold of 1.The mass threshold was based on similar AST duplication implementations [SR15], [Cod18] for java projects.

Type 2 clones:

For the type 2 clone detection we use the same parameters as type 1 but also normalize the structure. While traversing the AST we replace the values of variables, types, literals and functions with placeholder values.

Type 3 clones:

Type 3 clones are detected using the type 2 approach with a different *SimilarityThreshold*. While there is no general consensus [Kos08] on the optimal similarity threshold we settled for a threshold of 0.8 after reviewing several sources [MF], [SS15], [Clo18].

4 Clone visualization

This section describes our complete clone visualization tool which includes two types of visualizations: a clone browser and an edge bundle diagram. We built this tool as an Android app which makes it possible to do some analyzing of a code base without having access to a computer. This satisfies requirement 3 since many people have access to a mobile phone. We also host the results remotely using a Github page. This makes it convenient to run an analysis overnight and publish the results automatically to all Android clients and thus satisfying requirement 4.

4.1 Clone browser

The clone browser is implemented using Android. The goal of the browser is to give the user a tool they need for creating a mental model of a certain clone segment, using a bottom-up approach. Therefore, the clone browser satisfies requirement 1.

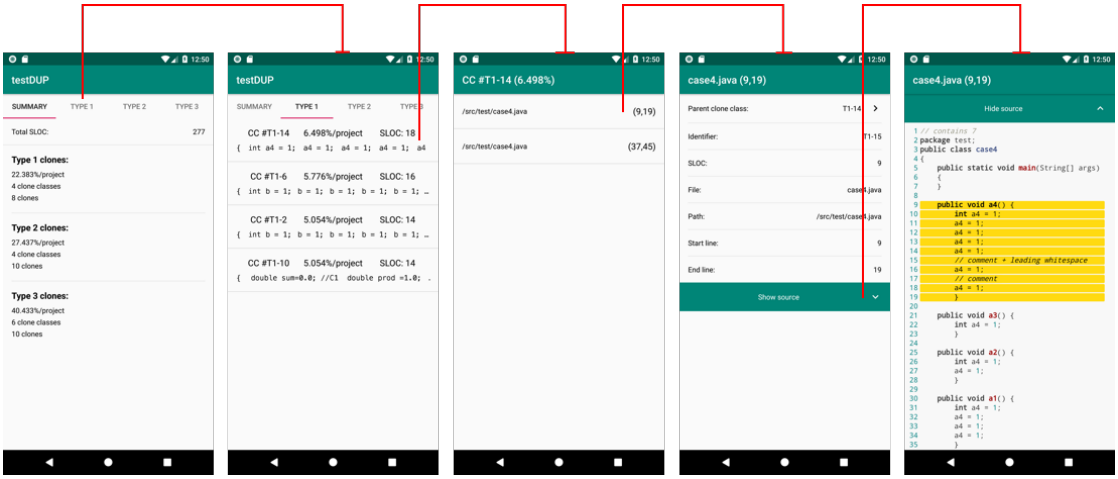

Figure 1: Clone browser UI

As can be seen in figure 1, a user is presented with an overview of the project they selected to review. Here some general statistics can be found about the project and the various types of clones. Selecting a type of clone in the list will bring the user to the edge bundle diagram. Switching tabs will result in a list view of all clone classes for that specific type. This separation makes sure that requirement 5 is satisfied. The list of clone classes is sorted on size and clicking a clone class will navigate the user to the list of clones within that clone class. After clicking an individual clone, the clone's details are shown and it's possible to collapse a syntax highlighted code view of the clone's source file. In this source file, the actual clone segment is highlighted.

4.2 Clone diagram

For the visualization of clones in a project we use an inverted radial layout 2 to visualize the code duplication structure within a project. Relations between duplicated segments are visualized using edge bundles [Hol06] [HBJ12] making it convenient to interpret the current situation. Users can both rearrange the diagram dynamically or switch to the clone browser by selecting duplicate elements in the diagram. This implementation satisfies requirement 2.

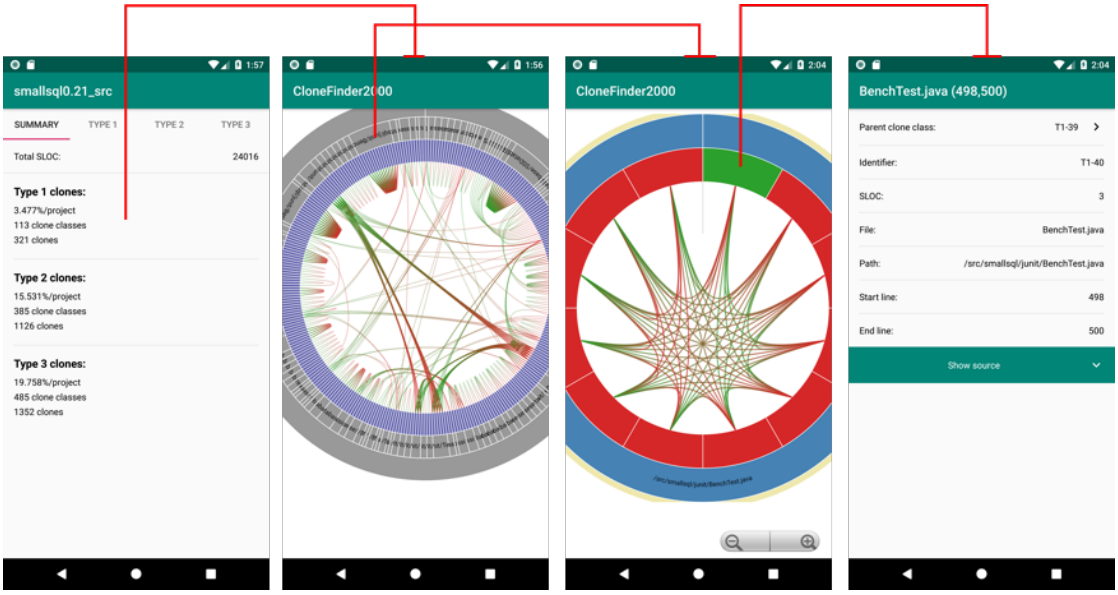

Figure 2: Example of our app analysing the smallsql source.

The inner ring of the diagram shows the clones. Besides drawing relations from these clones we also made these close double-clickable for further navigation into the details of a clone (just like the browser). The middle ring contains the nodes for every file in the code base that contains clone segments. It's also possible to select a file and zoom in on the clones within that file. This is helpful for analyzing a single source that contains a lot of clones by itself.

5 Test framework

This section describes how we applied automated testing to verify the correctness of our implementation.

5.1 Clone verification

For the verification of our clone detector we implemented the 11 type 1, 2 and 3 scenario's used in *Scenario-Based Comparison of Clone Detection Techniques* [RC08].
The results of testing failures are written to json files for manual inspection.

5.1.1 Type 1 verification:

To verify our type 1 results our framework verifies that the type 1 scenarios:

- are type 1, 2 and 3 clones of the original copy.
- are type 1, 2 and 3 clones of the other type 1 scenarios.

```
public class OriginalCopy {
    void sumProd(int n) {
    double sum=0.0; //C1
    double prod =1.0;
    for (int i=1; i<=n; i++)
      {sum=sum + i;
      prod = prod * i;
      foo(sum, prod); }}
}
```

Original

```
public class CopyOneA {
    void sumProd(int n) {
    double sum=0.0; //C1
    double prod =1.0;
      for (int i=1; i<=n; i++)
        {sum=sum + i;
        prod = prod * i;
        foo(sum, prod); }}
}
```

T1 scenario A

```
public class CopyOneB {
    void sumProd(int n) {
    double sum=0.0; //C1'
    double prod =1.0; //C
    for (int i=1; i<=n; i++)
      {sum=sum + i;
      prod = prod * i;
      foo(sum, prod); }}
}
```

T1 scenario B

Figure 3: Examples of the type 1 scenarios.

5.1.2 Type 2 verification:

To verify our type 2 results our framework verifies that the type 2 scenarios:

- are **not** type 1 clones of the original copy.
- are **not** type 1 clones of the other type 2 scenarios.
- are type 2 and 3 clones of the original copy.
- are type 2 and 3 clones of the other type 2 scenarios.

```
public class OriginalCopy {
    void sumProd(int n) {
    double sum=0.0; //C1
    double prod =1.0;
    for (int i=1; i<=n; i++)
      {sum=sum + i;
      prod = prod * i;
      foo(sum, prod); }}
}
```

Original

```
public class CopyTwoA {
    void sumProd(int n){
      double s=0.0; //C1
      double p =1.0;
      for (int j=1; j<=n; j++)
      {s=s + j;
      p = p * j;
      foo(s, p); }}
}
```

T2 scenario A

```
public class CopyTwoB {
    void sumProd(int n){
      double s=0.0; //C1
      double p =1.0;
      for (int j=1; j<=n; j++)
      {s=s + j;
      p = p * j;
      foo(p, s); }}
}
```

T2 scenario B

Figure 4: Examples of the type 2 scenarios.

5.1.3 Type 3 verification:

To verify our type 3 results our framework verifies that the type 3 scenarios:

- are **not** type 1 and 2 clones of the original copy.
- are **not** type 1 and 2 clones of the other type 3 scenarios.
- are type 3 clones of the original copy.

- are type 3 clones of the other type 3 scenarios.

```
1 public class OriginalCopy {
      void sumProd(int n) {
3     double sum=0.0; //C1
      double prod =1.0;
5     for (int i=1; i<=n; i++)
        {sum=sum + i;
7       prod = prod * i;
        foo(sum, prod); }}
9 }
```

Original

```
1 public class CopyThreeA {
      void sumProd(int n) {
3       double sum=0.0; //C1
        double prod =1.0;
5       for (int i=1; i<=n; i++)
          {sum=sum + i;
7         prod = prod * i;
          foo(sum, prod, n); }}
9 }
```

T3 scenario A

```
1 public class CopyThreeB {
      void sumProd(int n) {
3       double sum=0.0; //C1
        double prod =1.0;
5       for (int i=1; i<=n; i++)
          {sum=sum + i;
7         prod = prod * i;
          foo(prod); }}
9 }
```

T3 scenario B

Figure 5: Examples of the type 3 scenarios.

6 Threads to validity

Our aim was to create a tool to explore and analyze code duplication for a platform with limited interaction and presentation capabilities. The visualization fails in scenarios with a large number of clones present in a project creating a seemingly random noise of patterns on a small screen. This counters our original intent of presenting the clones in a visually pleasing and intuitive manner.
A possible solution could be to add weights to clone relations and only show the relations based on a set of thresholds in a single view.

Another potential issue is that we currently assume a flattened project structure. A potential source of duplication is the inclusion of overlapping and/or duplicate libraries within a project. Our current implementation does not present such scenarios intuitively and expects the user to find it out by their selves.

A final thread to the validity of our type 3 clone finding is the hashed bucket mechanism. Because the hashing does not take the similarity threshold into account while bucketing. It can happen that two potential type 3 clones are never in the same bucket and thus never compared with each other.

7 Results

This section contains the summarized results for every project and type.

7.1 smallSQL

We measured the following metrics on the smallSQL project:

	Percentage:	Clones:	Clone classes:
Type 1:	1.158%/project	59	28
Type 2:	6.829%/project	134	52
Type 3:	6.829%/project	134	52

Metrics of smallSQL

7.2 hsqldb

We measured the following metrics on the hsqldb project:

	Percentage:	Clones:	Clone classes:
Type 1:	4.197%/project	870	361
Type 2:	7.746%/project	983	441
Type 3:	7.746%/project	983	441

Metrics of Hsqldb

8 Example results

This section contains some example results per type and project.

8.1 SmallSQL - type 1

```
if(status.nodeValue != null){
            if(status.nodeValue instanceof IndexNode){
              level++;
              nodeStack.push(
                new IndexNodeScrollStatus(  (IndexNode)status.nodeValue,
                            (expressions.get(level).getAlias() != SQLTokenizer.DESC_STR),
                            scroll, level));
              continue;
            }else
              return getReturnValue(status.nodeValue);
          }
          //There is no RowOffset in this node
```

Biggest clone (11 SLOC)

```
/**Clones were found in
 * /src/smallsql/junit/TestMoneyRounding.java,
 * /src/smallsql/junit/TestOperatoren.java,
 * /src/smallsql/junit/TestDataTypes.java,
 * /src/smallsql/junit/TestFunctions.java.
 */
{
        try{
            Connection con = AllTests.getConnection();
            Statement st = con.createStatement();
            st.execute("drop table " + table);
            st.close();
        }catch(Throwable e){
            //e.printStackTrace();
        }
    }
```

Biggest clone class by total SLOC;example clone (36 total SLOC)

```
// Location: /src/smallsql/junit/TestFunctions.java
{
        try{
            Connection con = AllTests.getConnection();
            Statement st = con.createStatement();
            st.execute("drop table " + table);
            st.close();
        }catch(Throwable e){
            //e.printStackTrace();
        }
    }
```

Example 1/4

```
// Location: /src/smallsql/junit/TestDataTypes.java
{
        try{
            Connection con = AllTests.getConnection();
            Statement st = con.createStatement();
            st.execute("drop table " + table);
            st.close();
        }catch(Throwable e){
            //e.printStackTrace();
        }
    }
```

Example 2/4

```
// Location: /src/smallsql/junit/TestOperatoren.java
{
        try{
            Connection con = AllTests.getConnection();
            Statement st = con.createStatement();
            st.execute("drop table " + table);
            st.close();
        }catch(Throwable e){
            //e.printStackTrace();
        }
    }
```

Example 3/4

```
// Location: /src/smallsql/junit/TestMoneyRounding.java
{
        try{
            Connection con = AllTests.getConnection();
            Statement st = con.createStatement();
            st.execute("drop table " + table);
            st.close();
        }catch(Throwable e){
            //e.printStackTrace();
        }
    }
```

Example 4/4

8.2 SmallSQL - type 2

```
// /src/smallsql/database/language/Language_it.java
package smallsql.database.language;

/**
 * Extended localization class for Italian language.
 */
public class Language_it extends Language {
```

```
  protected Language_it() {
    addMessages(ENTRIES);
  }

  public String[][] getEntries() {
    return ENTRIES;
  }

  ////////////////////////////////////////////////////////////////////////
  // MESSAGES
  ////////////////////////////////////////////////////////////////////////

  private final String[][] ENTRIES = {
{ UNSUPPORTED_OPERATION           , "Operazione non supportata: {0}." },
{ CANT_LOCK_FILE                  , "Impossibile bloccare il file ''{0}''. Un database SmallSQL Database pu
    essere aperto da un unico processo." },

{ DB_EXISTENT                     , "Il database ''{0}''    gi  esistente." },
{ DB_NONEXISTENT                  , "Il database ''{0}'' Non esiste." },
{ DB_NOT_DIRECTORY                , "La directory ''{0}'' non   un database SmallSQL." },
{ DB_NOTCONNECTED                 , "L''utente non   connesso a un database." },

{ CONNECTION_CLOSED               , "La connessione    gi  chiusa." },

{ VIEW_INSERT                     , "INSERT non   supportato per una view." },
{ VIEWDROP_NOT_VIEW               , "Non   possibile effettuare DROP VIEW con ''{0}'' perch  non   una
    view." },
{ VIEW_CANTDROP                   , "Non si pu  effettuare drop sulla view ''{0}''." },

{ RSET_NOT_PRODUCED               , "Nessun ResultSet   stato prodotto." },
{ RSET_READONLY                   , "Il ResultSet   di sola lettura." },
{ RSET_FWDONLY                    , "Il ResultSet   forward only." }, // no real translation
{ RSET_CLOSED                     , "Il ResultSet   chiuso." },
{ RSET_NOT_INSERT_ROW             , "Il cursore non   attualmente nella riga ''InsertRow''." },
{ RSET_ON_INSERT_ROW              , "Il cursore   attualmente nella riga ''InsertRow''." },
{ ROWSOURCE_READONLY              , "Il Rowsource   di sola lettura." },

{ STMT_IS_CLOSED                  , "Lo Statement   in stato chiuso." },

{ SUBQUERY_COL_COUNT              , "Il conteggio delle colonne nella subquery deve essere 1 e non {0}." },
{ JOIN_DELETE                     , "DeleteRow non supportato nelle join." },
{ JOIN_INSERT                     , "InsertRow non supportato nelle join." },
{ DELETE_WO_FROM                  , "DeleteRow necessita un''espressione FROM." },
{ INSERT_WO_FROM                  , "InsertRow necessita un''espressione FROM." },

{ TABLE_CANT_RENAME               , "La tabella ''{0}'' non pu  essere rinominata." },
{ TABLE_CANT_DROP                 , "Non si pu  effettuare DROP della tabella ''{0}''." },
{ TABLE_CANT_DROP_LOCKED          , "Non si pu  effettuare DROP della tabella ''{0}'' perch    in LOCK."
    },
{ TABLE_CORRUPT_PAGE              , "Pagina della tabella corrotta alla posizione: {0}." },
{ TABLE_MODIFIED                  , "La tabella ''{0}''   stata modificata." },
{ TABLE_DEADLOCK                  , "Deadlock: non si pu  mettere un lock sulla tabella ''{0}''." },
{ TABLE_OR_VIEW_MISSING           , "La tabella/view ''{0}'' non esiste." },
{ TABLE_FILE_INVALID              , "Il file ''{0}'' non include una tabella SmallSQL valida." },
```

```
{ TABLE_OR_VIEW_FILE_INVALID      , "Il file ''{0}'' non    un contenitore valido di tabella/view." },
{ TABLE_EXISTENT                  , "La tabella/vista ''{0}''     gi  esistente." },

{ FK_NOT_TABLE                    , "''{0}'' non    una tabella." },
{ PK_ONLYONE                      , "Una tabella pu  avere solo una primary key." },
{ KEY_DUPLICATE                   , "Chiave duplicata." },

{ MONTH_TOOLARGE                  , "Valore del mese troppo alto del in DATE o TIMESTAMP ''{0}''." },
{ DAYS_TOOLARGE                   , "Valore del giorno troppo altro in DATE o TIMESTAMP ''{0}''." },
{ HOURS_TOOLARGE                  , "Valore delle ore troppo alto in in TIME o TIMESTAMP ''{0}''." },
{ MINUTES_TOOLARGE                , "Valore dei minuti troppo alto in TIME o TIMESTAMP ''{0}''." },
{ SECS_TOOLARGE                   , "Valore dei secondi troppo alto in TIME o TIMESTAMP ''{0}''." },
{ MILLIS_TOOLARGE                 , "VAlore dei millisecondi troppo alto in TIMESTAMP ''{0}''." },
{ DATETIME_INVALID                , "''{0}''    un DATE, TIME or TIMESTAMP non valido." },

{ UNSUPPORTED_CONVERSION_OPER     , "Conversione non supportata verso il tipo di dato ''{0}'' dal tipo
    ''{1}'' per l''operazione ''{2}''." },
{ UNSUPPORTED_DATATYPE_OPER       , "Tipo di dato ''{0}'' non supportato per l''operazione ''{1}''." },
{ UNSUPPORTED_DATATYPE_FUNC       , "Tipo di dato ''{0}'' non supportato per la funzione ''{1}''." },
{ UNSUPPORTED_CONVERSION_FUNC     , "Conversione verso il tipo di dato ''{0}'' non supportato per la funzione
     ''{1}''." },
{ UNSUPPORTED_TYPE_CONV           , "Tipo non supportato per la funzione CONVERT: {0}." },
{ UNSUPPORTED_TYPE_SUM            , "Tipo non supportato per la funzione SUM: ''{0}''." },
{ UNSUPPORTED_TYPE_MAX            , "Tipo non supportato per la funzione MAX: ''{0}''." },
{ UNSUPPORTED_CONVERSION          , "Non    possible convertire ''{0}'' [{1}] in ''{2}''." },
{ INSERT_INVALID_LEN              , "Lunghezza non valida ''{0}'' per la funzione INSERT." },
{ SUBSTR_INVALID_LEN              , "Lunghezza non valida ''{0}'' per la funzione SUBSTRING." },

{ VALUE_STR_TOOLARGE              , "Stringa troppo lunga per la colonna." },
{ VALUE_BIN_TOOLARGE              , "Valore binario di lunghezza {0} eccessiva per la colonna di lunghezza
    {1}." },
{ VALUE_NULL_INVALID              , "Valori nulli non validi per la colonna ''{0}''." },
{ VALUE_CANT_CONVERT              , "Impossible convertire un valore {0} in un valore {1}." },

{ BYTEARR_INVALID_SIZE            , "Lunghezza non valida per un array di bytes: {0}." },
{ LOB_DELETED                     , "L''oggetto LOB    stato cancellato." },

{ PARAM_CLASS_UNKNOWN             , "Classe sconosciuta (''{0}'') per il parametro." },
{ PARAM_EMPTY                     , "Il parametro {0}    vuoto." },
{ PARAM_IDX_OUT_RANGE             , "L''indice {0} per il parametro    fuori dall''intervallo consentito ( 1
    <= n <= {1} )." },

{ COL_DUPLICATE                   , "Nome di colonna duplicato: ''{0}''." },
{ COL_MISSING                     , "Colonna ''{0}'' non trovata." },
{ COL_VAL_UNMATCH                 , "Il conteggio di colonne e valori non    identico." },
...
```

Biggest clone (122 SLOC);part of biggest clone class (244 total SLOC)

```
// Location: /src/smallsql/database/ExpressionFunctionYear.java
{
    if(param1.isNull()) return 0;
    DateTime.Details details = new DateTime.Details(param1.getLong());
    return details.year;
```

```
  }
```

Example 1/4

```
// Location: /src/smallsql/database/ExpressionFunctionMinute.java
{
    if(param1.isNull()) return 0;
    DateTime.Details details = new DateTime.Details(param1.getLong());
    return details.minute;
  }
```

Example 2/4

```
// Location: /src/smallsql/database/ExpressionFunctionDayOfMonth.java
{
    if(param1.isNull()) return 0;
    DateTime.Details details = new DateTime.Details(param1.getLong());
    return details.day;
  }
```

Example 3/4

```
// Location: /src/smallsql/database/ExpressionFunctionHour.java
{
    if(param1.isNull()) return 0;
    DateTime.Details details = new DateTime.Details(param1.getLong());
    return details.hour;
  }
```

Example 4/4

8.3 SmallSQL - type 3

```
// /src/smallsql/database/language/Language_de.java
package smallsql.database.language;

/**
 * Extended localization class for German language.
 */
public class Language_de extends Language {
  protected Language_de() {
    addMessages(ENTRIES);
  }

  public String[][] getEntries() {
    return ENTRIES;
  }

  ////////////////////////////////////////////////////////////////////////
  // MESSAGES
  ////////////////////////////////////////////////////////////////////////
```

```
20      private final String[][] ENTRIES = {
                { UNSUPPORTED_OPERATION           , "Nicht unterst tzte Funktion: {0}" },
22              { CANT_LOCK_FILE                  , "Die Datei ''{0}'' kann nicht gelockt werden. Eine einzelne
        SmallSQL Datenbank kann nur f r einen einzigen Prozess ge ffnet werden." },

24              { DB_EXISTENT                     , "Die Datenbank ''{0}'' existiert bereits." },
                { DB_NONEXISTENT                  , "Die Datenbank ''{0}'' existiert nicht." },
26              { DB_NOT_DIRECTORY                , "Das Verzeichnis ''{0}'' ist keine SmallSQL Datenbank." },
                { DB_NOTCONNECTED                 , "Sie sind nicht mit einer Datenbank verbunden." },
28
                { CONNECTION_CLOSED               , "Die Verbindung ist bereits geschlossen." },
30
                { VIEW_INSERT                     , "INSERT wird nicht unterst tzt f r eine View." },
32              { VIEWDROP_NOT_VIEW               , "DROP VIEW kann nicht mit ''{0}'' verwendet werden, weil es
        keine View ist." },
                { VIEW_CANTDROP                   , "View ''{0}'' kann nicht gel scht werden." },
34
                { RSET_NOT_PRODUCED               , "Es wurde kein ResultSet erzeugt." },
36              { RSET_READONLY                   , "Das ResultSet ist schreibgesch tzt." },
                { RSET_FWDONLY                    , "Das ResultSet ist forward only." },
38              { RSET_CLOSED                     , "Das ResultSet ist geschlossen." },
                { RSET_NOT_INSERT_ROW             , "Der Cursor zeigt aktuell nicht auf die Einf geposition (
        insert row)." },
40              { RSET_ON_INSERT_ROW              , "Der Cursor zeigt aktuell auf die Einf geposition (insert
        row)." },
                { ROWSOURCE_READONLY              , "Die Rowsource ist schreibgesch tzt." },
42              { STMT_IS_CLOSED                  , "Das Statement ist bereits geschlossen." },

44              { SUBQUERY_COL_COUNT              , "Die Anzahl der Spalten in der Subquery muss 1 sein und nicht
         {0}." },
                { JOIN_DELETE                     , "Die Methode deleteRow wird nicht unterst tzt f r Joins."
        },
46              { JOIN_INSERT                     , "Die Methode insertRow wird nicht unterst tzt f r Joins."
        },
                { DELETE_WO_FROM                  , "Die Methode deleteRow ben tigt einen FROM Ausdruck." },
48              { INSERT_WO_FROM                  , "Die Methode insertRow ben tigt einen FROM Ausdruck." },

50              { TABLE_CANT_RENAME               , "Die Tabelle ''{0}'' kann nicht umbenannt werden." },
                { TABLE_CANT_DROP                 , "Die Tabelle ''{0}'' kann nicht gel scht werden." },
52              { TABLE_CANT_DROP_LOCKED          , "Die Tabelle ''{0}'' kann nicht gel scht werden, weil sie
        gelockt ist." },
                { TABLE_CORRUPT_PAGE              , "Besch digte Tabellenseite bei Position: {0}." },
54              { TABLE_MODIFIED                  , "Die Tabelle ''{0}'' wurde modifiziert." },
                { TABLE_DEADLOCK                  , "Deadlock, es kann kein Lock erzeugt werden f r Tabelle
        ''{0}''." },
56              { TABLE_OR_VIEW_MISSING           , "Tabelle oder View ''{0}'' existiert nicht." },
                { TABLE_FILE_INVALID              , "Die Datei ''{0}'' enth lt keine g ltige SmallSQL Tabelle."
         },
58              { TABLE_OR_VIEW_FILE_INVALID      , "Die Datei ''{0}'' ist keine g ltiger Tabellen oder View
        Speicher." },
                { TABLE_EXISTENT                  , "Die Tabelle oder View ''{0}'' existiert bereits." },
60
                { FK_NOT_TABLE                    , "''{0}'' ist keine Tabelle." },
62              { PK_ONLYONE                      , "Eine Tabelle kann nur einen Prim rschl ssel haben." },
```

```
            { KEY_DUPLICATE                   , "Doppelter Schl ssel." },
64
            { MONTH_TOOLARGE                  , "Der Monat ist zu gro  im DATE oder TIMESTAMP Wert ''{0}''."
     },
66          { DAYS_TOOLARGE                   , "Die Tage sind zu gro  im DATE oder TIMESTAMP Wert ''{0}''."
     },
            { HOURS_TOOLARGE                  , "Die Stunden sind zu gro  im TIME oder TIMESTAMP Wert
    ''{0}''." },
68          { MINUTES_TOOLARGE                , "Die Minuten sind zu gro  im TIME oder TIMESTAMP Wert
    ''{0}''." },
            { SECS_TOOLARGE                   , "Die Sekunden sind zu gro  im TIME oder TIMESTAMP Wert
    ''{0}''." },
70          { MILLIS_TOOLARGE                 , "Die Millisekunden sind zu gro  im TIMESTAMP Wert ''{0}''."
    },
            { DATETIME_INVALID                , "''{0}'' ist ein ung ltiges DATE, TIME or TIMESTAMP." },
72
            { UNSUPPORTED_CONVERSION_OPER     , "Nicht unterst tzte Konvertierung zu Datentyp ''{0}'' von
    Datentyp ''{1}'' f r die Operation ''{2}''." },
74          { UNSUPPORTED_DATATYPE_OPER       , "Nicht unterst tzter Datentyp ''{0}'' f r Operation
    ''{1}''." },
            { UNSUPPORTED_DATATYPE_FUNC       , "Nicht unterst tzter Datentyp ''{0}'' f r Funktion ''{1}''.
    " },
76          { UNSUPPORTED_CONVERSION_FUNC     , "Nicht unterst tzte Konvertierung zu Datentyp ''{0}'' f r
    Funktion ''{1}''." },
            { UNSUPPORTED_TYPE_CONV           , "Nicht unterst tzter Typ f r CONVERT Funktion: {0}." },
78          { UNSUPPORTED_TYPE_SUM            , "Nicht unterst tzter Datentyp ''{0}'' f r SUM Funktion." },
            { UNSUPPORTED_TYPE_MAX            , "Nicht unterst tzter Datentyp ''{0}'' f r MAX Funktion." },
80          { UNSUPPORTED_CONVERSION          , "Kann nicht konvertieren ''{0}'' [{1}] zu ''{2}''." },
            { INSERT_INVALID_LEN              , "Ung ltige L nge ''{0}'' in Funktion INSERT." },
82          { SUBSTR_INVALID_LEN              , "Ung ltige L nge ''{0}'' in Funktion SUBSTRING." },

84          { VALUE_STR_TOOLARGE              , "Der String Wert ist zu gro  f r die Spalte." },
            { VALUE_BIN_TOOLARGE              , "Ein Bin re Wert mit L nge {0} ist zu gro  f r eine
    Spalte mit der Gr  e {1}." },
86          { VALUE_NULL_INVALID              , "Null Werte sind ung ltig f r die Spalte ''{0}''." },
            { VALUE_CANT_CONVERT              , "Kann nicht konvertieren ein {0} Wert zu einem {1} Wert." },
88
            { BYTEARR_INVALID_SIZE            , "Ung ltige Bytearray Gro e {0} f r UNIQUEIDENFIER." },
90          { LOB_DELETED                     , "Lob Objekt wurde gel scht." },

92          { PARAM_CLASS_UNKNOWN             , "Unbekante Parameter Klasse: ''{0}''." },
            { PARAM_EMPTY                     , "Parameter {0} ist leer." },
94          { PARAM_IDX_OUT_RANGE             , "Parameter Index {0} liegt au erhalb des
    G ltigkeitsbereiches. Der Wert muss zwischen 1 und {1} liegen." },

96          { COL_DUPLICATE                   , "Es gibt einen doppelten Spaltennamen: ''{0}''." },
            { COL_MISSING                     , "Spalte ''{0}'' wurde nicht gefunden." },
98          { COL_VAL_UNMATCH                 , "Die Spaltenanzahl und Werteanzahl ist nicht identisch." },
            { COL_INVALID_SIZE                , "Ung ltige Spaltengr  e {0} f r Spalte ''{1}''." },
100 ...
```

Biggest clone (122 SLOC);part of biggest clone class (244 total SLOC)

```
// Location: /src/smallsql/database/ExpressionFunctionPower.java
```

```
package smallsql.database;


final class ExpressionFunctionPower extends ExpressionFunctionReturnFloat {

    final int getFunction(){ return SQLTokenizer.POWER; }

    boolean isNull() throws Exception{
        return param1.isNull() || param2.isNull();
    }

    final double getDouble() throws Exception{
    if(isNull()) return 0;
        return Math.pow( param1.getDouble(), param2.getDouble() );
    }
}
```

Example 1/2

```
// Location: /src/smallsql/database/ExpressionFunctionATan2.java
package smallsql.database;


final class ExpressionFunctionATan2 extends ExpressionFunctionReturnFloat {

    final int getFunction(){ return SQLTokenizer.ATAN2; }

    boolean isNull() throws Exception{
        return param1.isNull() || param2.isNull();
    }

    final double getDouble() throws Exception{
    if(isNull()) return 0;
        return Math.atan2( param1.getDouble(), param2.getDouble() );
    }
}
```

Example 2/2

8.4 Hsqldb - type 1

```
// Location /hsqldb/src/org/hsqldb/dbinfo/DatabaseInformationFull.java
// common type block
                if (type.isCharacterType()) {
                    row[character_maximum_length] =
                        ValuePool.getLong(type.precision);
                    row[character_octet_length] =
                        ValuePool.getLong(type.precision * 2);
                    row[character_set_catalog] =
                        database.getCatalogName().name;
                    row[character_set_schema] =
                        ((CharacterType) type).getCharacterSet()
```

```
                        .getSchemaName().name;
                row[character_set_name] =
                    ((CharacterType) type).getCharacterSet().getName()
                        .name;
                row[collation_catalog] = database.getCatalogName().name;
                row[collation_schema] =
                    ((CharacterType) type).getCollation().getSchemaName()
                        .name;
                row[collation_name] =
                    ((CharacterType) type).getCollation().getName().name;
            } else if (type.isNumberType()) {
                row[numeric_precision] = ValuePool.getLong(
                    ((NumberType) type).getNumericPrecisionInRadix());
                row[declared_numeric_precision] = ValuePool.getLong(
                    ((NumberType) type).getNumericPrecisionInRadix());

                if (type.isExactNumberType()) {
                    row[numeric_scale] = row[declared_numeric_scale] =
                        ValuePool.getLong(type.scale);
                }

                row[numeric_precision_radix] =
                    ValuePool.getLong(type.getPrecisionRadix());
            } else if (type.isBooleanType()) {

                //
            } else if (type.isDateTimeType()) {
                row[datetime_precision] = ValuePool.getLong(type.scale);
            } else if (type.isIntervalType()) {
                row[data_type] = "INTERVAL";
                row[interval_type] =
                    ((IntervalType) type).getQualifier(type.typeCode);
                row[interval_precision] =
                    ValuePool.getLong(type.precision);
                row[datetime_precision] = ValuePool.getLong(type.scale);
            } else if (type.isBinaryType()) {
                row[character_maximum_length] =
                    ValuePool.getLong(type.precision);
                row[character_octet_length] =
                    ValuePool.getLong(type.precision);
            } else if (type.isBitType()) {
                row[character_maximum_length] =
                    ValuePool.getLong(type.precision);
                row[character_octet_length] =
                    ValuePool.getLong(type.precision);
            } else if (type.isArrayType()) {
                row[maximum_cardinality] =
                    ValuePool.getLong(type.arrayLimitCardinality());
                row[data_type] = "ARRAY";
            }
```

Biggest clone (55 SLOC)

```
// Location /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java
```

```
catch (SQLException se) { junit.framework.AssertionFailedError ase
                = new junit.framework.AssertionFailedError(se.getMessage());
            ase.initCause(se);
            throw ase;
        }

// Clone class locations
// /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java, /hsqldb/
     src/org/hsqldb/test/TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/TestOdbcService.java, /hsqldb/src/
     org/hsqldb/test/TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java, /hsqldb/src/org/
     hsqldb/test/TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java, /hsqldb/src/org/hsqldb/
     test/TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/
     TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/
     TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/
     TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/
     TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/
     TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/
     AbstractTestOdbc.java, /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/
     TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/
     TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/
     TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/TestOdbcService.java, /hsqldb/src/org/hsqldb/test/
     TestOdbcService.java, /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/
     TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/
     TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/TestOdbcService.java, /hsqldb/src/org/hsqldb/test/
     TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/
     TestOdbcService.java, /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/
     TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/
     TestOdbcTypes.java, /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java
```

Biggest clone class by total SLOC;example clone (251 total SLOC)

```
// Location /hsqldb/src/org/hsqldb/NumberSequence.java
switch (dataType.typeCode) {

            case Types.TINYINT :
                max = Byte.MAX_VALUE;
                min = Byte.MIN_VALUE;
                break;

            case Types.SQL_SMALLINT :
                max = Short.MAX_VALUE;
                min = Short.MIN_VALUE;
                break;

            case Types.SQL_INTEGER :
                max = Integer.MAX_VALUE;
                min = Integer.MIN_VALUE;
                break;

            case Types.SQL_BIGINT :
                max = Long.MAX_VALUE;
                min = Long.MIN_VALUE;
                break;

```

```
            case Types.SQL_NUMERIC :
            case Types.SQL_DECIMAL :
                max = Long.MAX_VALUE;
                min = Long.MIN_VALUE;
                break;

            default :
                throw Error.runtimeError(ErrorCode.U_S0500, "NumberSequence");
        }
```

Example 1/3

```
// /hsqldb/src/org/hsqldb/NumberSequence.java
switch (dataType.typeCode) {

            case Types.TINYINT :
                max = Byte.MAX_VALUE;
                min = Byte.MIN_VALUE;
                break;

            case Types.SQL_SMALLINT :
                max = Short.MAX_VALUE;
                min = Short.MIN_VALUE;
                break;

            case Types.SQL_INTEGER :
                max = Integer.MAX_VALUE;
                min = Integer.MIN_VALUE;
                break;

            case Types.SQL_BIGINT :
                max = Long.MAX_VALUE;
                min = Long.MIN_VALUE;
                break;

            case Types.SQL_NUMERIC :
            case Types.SQL_DECIMAL :
                max = Long.MAX_VALUE;
                min = Long.MIN_VALUE;
                break;

            default :
                throw Error.runtimeError(ErrorCode.U_S0500, "NumberSequence");
        }
```

Example 2/3

```
// Location: /hsqldb/src/org/hsqldb/NumberSequence.java
switch (dataType.typeCode) {

            case Types.TINYINT :
                max = Byte.MAX_VALUE;
                min = Byte.MIN_VALUE;
                break;
```

```
8
            case Types.SQL_SMALLINT :
10              max = Short.MAX_VALUE;
                min = Short.MIN_VALUE;
12              break;

14          case Types.SQL_INTEGER :
                max = Integer.MAX_VALUE;
16              min = Integer.MIN_VALUE;
                break;
18
            case Types.SQL_BIGINT :
20              max = Long.MAX_VALUE;
                min = Long.MIN_VALUE;
22              break;

24          case Types.SQL_NUMERIC :
            case Types.SQL_DECIMAL :
26              max = Long.MAX_VALUE;
                min = Long.MIN_VALUE;
28              break;

30          default :
                throw Error.runtimeError(ErrorCode.U_S0500, "NumberSequence");
32      }
```

Example 3/3

8.5 Hsqldb - type 2

```
// Location: /hsqldb/src/org/hsqldb/dbinfo/DatabaseInformationMain.java (3209, 3312)
2 {

4       Table t = sysTables[SYSTEM_SEQUENCES];

6       if (t == null) {
            t = createBlankTable(sysTableHsqlNames[SYSTEM_SEQUENCES]);
8
            addColumn(t, "SEQUENCE_CATALOG", SQL_IDENTIFIER);
10          addColumn(t, "SEQUENCE_SCHEMA", SQL_IDENTIFIER);
            addColumn(t, "SEQUENCE_NAME", SQL_IDENTIFIER);
12          addColumn(t, "DATA_TYPE", CHARACTER_DATA);
            addColumn(t, "NUMERIC_PRECISION", CARDINAL_NUMBER);
14          addColumn(t, "NUMERIC_PRECISION_RADIX", CARDINAL_NUMBER);
            addColumn(t, "NUMERIC_SCALE", CARDINAL_NUMBER);
16          addColumn(t, "MAXIMUM_VALUE", CHARACTER_DATA);
            addColumn(t, "MINIMUM_VALUE", CHARACTER_DATA);
18          addColumn(t, "INCREMENT", CHARACTER_DATA);
            addColumn(t, "CYCLE_OPTION", YES_OR_NO);
20          addColumn(t, "DECLARED_DATA_TYPE", CHARACTER_DATA);
            addColumn(t, "DECLARED_NUMERIC_PRECISION", CARDINAL_NUMBER);
22          addColumn(t, "DECLARED_NUMERIC_SCALE", CARDINAL_NUMBER);
```

```
            // HSQLDB-specific
            addColumn(t, "START_WITH", CHARACTER_DATA);
            addColumn(t, "NEXT_VALUE", CHARACTER_DATA);

            // order SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME
            // false PK, as CATALOG may be null
            HsqlName name = HsqlNameManager.newInfoSchemaObjectName(
                sysTableHsqlNames[SYSTEM_SEQUENCES].name, false,
                SchemaObject.INDEX);

            t.createPrimaryKeyConstraint(name, new int[] {
                0, 1, 2
            }, false);

            return t;
        }

        //
        final int sequence_catalog           = 0;
        final int sequence_schema            = 1;
        final int sequence_name              = 2;
        final int data_type                  = 3;
        final int numeric_precision          = 4;
        final int numeric_precision_radix    = 5;
        final int numeric_scale              = 6;
        final int maximum_value              = 7;
        final int minimum_value              = 8;
        final int increment                  = 9;
        final int cycle_option               = 10;
        final int declared_data_type         = 11;
        final int declared_numeric_precision = 12;
        final int declared_numeric_scale     = 13;
        final int start_with                 = 14;
        final int next_value                 = 15;

        //
        Iterator       it;
        Object[]       row;
        NumberSequence sequence;

        it = database.schemaManager.databaseObjectIterator(
            SchemaObject.SEQUENCE);

        while (it.hasNext()) {
            sequence = (NumberSequence) it.next();

            if (!session.getGrantee().isAccessible(sequence)) {
                continue;
            }

            row = t.getEmptyRowData();

            NumberType type = (NumberType) sequence.getDataType();
```

```
            int radix =
78              (type.typeCode == Types.SQL_NUMERIC || type.typeCode == Types
                    .SQL_DECIMAL) ? 10
80                              : 2;

82          row[sequence_catalog] = database.getCatalogName().name;
            row[sequence_schema]  = sequence.getSchemaName().name;
84          row[sequence_name]    = sequence.getName().name;
            row[data_type]        = sequence.getDataType().getFullNameString();
86          row[numeric_precision] =
                ValuePool.getInt((int) type.getPrecision());
88          row[numeric_precision_radix]    = ValuePool.getInt(radix);
            row[numeric_scale]              = ValuePool.INTEGER_0;
90          row[maximum_value] = String.valueOf(sequence.getMaxValue());
            row[minimum_value] = String.valueOf(sequence.getMinValue());
92          row[increment] = String.valueOf(sequence.getIncrement());
            row[cycle_option]               = sequence.isCycle() ? "YES"
94                                                               : "NO";
            row[declared_data_type]         = row[data_type];
96          row[declared_numeric_precision] = row[numeric_precision];
            row[declared_numeric_scale]     = row[declared_numeric_scale];
98          row[start_with] = String.valueOf(sequence.getStartValue());
            row[next_value]                 = String.valueOf(sequence.peek());
100
            t.insertSys(session, store, row);
102     }

104     return t;
    }
```

Biggest clone (82 SLOC)

```
1   // Location /hsqldb/src/org/hsqldb/dbinfo/DatabaseInformationFull.java (2687,2746)
    // Other locations:
3   // /hsqldb/src/org/hsqldb/dbinfo/DatabaseInformationFull.java (5967, 6026)
    // /hsqldb/src/org/hsqldb/dbinfo/DatabaseInformationFull.java (3440, 3495)
5   // /hsqldb/src/org/hsqldb/dbinfo/DatabaseInformationFull.java (7932, 7985)
    // common type block
7               if (type.isCharacterType()) {
                    row[character_maximum_length] =
9                       ValuePool.getLong(type.precision);
                    row[character_octet_length] =
11                      ValuePool.getLong(type.precision * 2);
                    row[character_set_catalog] =
13                      database.getCatalogName().name;
                    row[character_set_schema] =
15                      ((CharacterType) type).getCharacterSet()
                            .getSchemaName().name;
17                  row[character_set_name] =
                        ((CharacterType) type).getCharacterSet().getName()
19                          .name;
                    row[collation_catalog] = database.getCatalogName().name;
21                  row[collation_schema] =
                        ((CharacterType) type).getCollation().getSchemaName()
```

```
                            .name;
                    row[collation_name] =
                        ((CharacterType) type).getCollation().getName().name;
                } else if (type.isNumberType()) {
                    row[numeric_precision] = ValuePool.getLong(
                        ((NumberType) type).getNumericPrecisionInRadix());
                    row[declared_numeric_precision] = ValuePool.getLong(
                        ((NumberType) type).getNumericPrecisionInRadix());

                    if (type.isExactNumberType()) {
                        row[numeric_scale] = row[declared_numeric_scale] =
                            ValuePool.getLong(type.scale);
                    }

                    row[numeric_precision_radix] =
                        ValuePool.getLong(type.getPrecisionRadix());
                } else if (type.isBooleanType()) {

                    //
                } else if (type.isDateTimeType()) {
                    row[datetime_precision] = ValuePool.getLong(type.scale);
                } else if (type.isIntervalType()) {
                    row[data_type] = "INTERVAL";
                    row[interval_type] =
                        ((IntervalType) type).getQualifier(type.typeCode);
                    row[interval_precision] =
                        ValuePool.getLong(type.precision);
                    row[datetime_precision] = ValuePool.getLong(type.scale);
                } else if (type.isBinaryType()) {
                    row[character_maximum_length] =
                        ValuePool.getLong(type.precision);
                    row[character_octet_length] =
                        ValuePool.getLong(type.precision);
                } else if (type.isBitType()) {
                    row[character_maximum_length] =
                        ValuePool.getLong(type.precision);
                    row[character_octet_length] =
                        ValuePool.getLong(type.precision);
                } else if (type.isArrayType()) {
                    row[maximum_cardinality] =
                        ValuePool.getLong(type.arrayLimitCardinality());
                    row[data_type] = "ARRAY";
                }
```

Biggest clone class by total SLOC example clone (212 total SLOC)

```
// Location: /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java (1297,1337)
{
        PreparedStatement ps = null;
        ResultSet rs = null;
        Time aTime = Time.valueOf("21:19:27");

        try {
            ps = netConn.prepareStatement(
```

```
                "INSERT INTO alltypes(id, t) VALUES(?, ?)");
            ps.setInt(1, 3);
            ps.setTime(2, aTime);
            assertEquals(1, ps.executeUpdate());
            ps.setInt(1, 4);
            assertEquals(1, ps.executeUpdate());
            ps.close();
            netConn.commit();
            ps = netConn.prepareStatement(
                "SELECT * FROM alltypes WHERE t = ?");
            ps.setTime(1, aTime);
            rs = ps.executeQuery();
            assertTrue("Got no rows with t = aTime", rs.next());
            assertEquals(Time.class, rs.getObject("t").getClass());
            assertTrue("Got only one row with t = aTime", rs.next());
            assertEquals(aTime, rs.getTime("t"));
            assertFalse("Got too many rows with t = aTime", rs.next());
        } catch (SQLException se) {
            junit.framework.AssertionFailedError ase
                = new junit.framework.AssertionFailedError(se.getMessage());
            ase.initCause(se);
            throw ase;
        } finally {
            try {
                if (rs != null) {
                    rs.close();
                }
                if (ps != null) {
                    ps.close();
                }
            } catch(Exception e) {
            }
        }
    }
```

Example 1/3

```
// Location: /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java (1386, 1426)
{
        PreparedStatement ps = null;
        ResultSet rs = null;
        Timestamp aTimestamp = Timestamp.valueOf("2009-03-27 17:18:19");

        try {
            ps = netConn.prepareStatement(
                "INSERT INTO alltypes(id, tsw) VALUES(?, ?)");
            ps.setInt(1, 3);
            ps.setTimestamp(2, aTimestamp);
            assertEquals(1, ps.executeUpdate());
            ps.setInt(1, 4);
            assertEquals(1, ps.executeUpdate());
            ps.close();
            netConn.commit();
            ps = netConn.prepareStatement(
```

```
                "SELECT * FROM alltypes WHERE tsw = ?");
            ps.setTimestamp(1, aTimestamp);
            rs = ps.executeQuery();
            assertTrue("Got no rows with tsw = aTimestamp", rs.next());
            assertEquals(Timestamp.class, rs.getObject("tsw").getClass());
            assertTrue("Got only one row with tsw = aTimestamp", rs.next());
            assertEquals(aTimestamp, rs.getTimestamp("tsw"));
            assertFalse("Got too many rows with tsw = aTimestamp", rs.next());
        } catch (SQLException se) {
            junit.framework.AssertionFailedError ase
                = new junit.framework.AssertionFailedError(se.getMessage());
            ase.initCause(se);
            throw ase;
        } finally {
            try {
                if (rs != null) {
                    rs.close();
                }
                if (ps != null) {
                    ps.close();
                }
            } catch(Exception e) {
            }
        }
    }
```

Example 2/3

```
// Location: /hsqldb/src/org/hsqldb/test/TestOdbcTypes.java (1344, 1384)
{
        PreparedStatement ps = null;
        ResultSet rs = null;
        Timestamp aTimestamp = Timestamp.valueOf("2009-03-27 17:18:19");

        try {
            ps = netConn.prepareStatement(
                "INSERT INTO alltypes(id, ts) VALUES(?, ?)");
            ps.setInt(1, 3);
            ps.setTimestamp(2, aTimestamp);
            assertEquals(1, ps.executeUpdate());
            ps.setInt(1, 4);
            assertEquals(1, ps.executeUpdate());
            ps.close();
            netConn.commit();
            ps = netConn.prepareStatement(
                "SELECT * FROM alltypes WHERE ts = ?");
            ps.setTimestamp(1, aTimestamp);
            rs = ps.executeQuery();
            assertTrue("Got no rows with ts = aTimestamp", rs.next());
            assertEquals(Timestamp.class, rs.getObject("ts").getClass());
            assertTrue("Got only one row with ts = aTimestamp", rs.next());
            assertEquals(aTimestamp, rs.getTimestamp("ts"));
            assertFalse("Got too many rows with ts = aTimestamp", rs.next());
        } catch (SQLException se) {
```

```
            junit.framework.AssertionFailedError ase
                = new junit.framework.AssertionFailedError(se.getMessage());
            ase.initCause(se);
            throw ase;
        } finally {
            try {
                if (rs != null) {
                    rs.close();
                }
                if (ps != null) {
                    ps.close();
                }
            } catch(Exception e) {
            }
        }
    }
```

Example 3/3

8.6 Hsqldb - type 3

```
// Location: /hsqldb/src/org/hsqldb/dbinfo/DatabaseInformationMain.java (3209,3312)
{

        Table t = sysTables[SYSTEM_SEQUENCES];

        if (t == null) {
            t = createBlankTable(sysTableHsqlNames[SYSTEM_SEQUENCES]);

            addColumn(t, "SEQUENCE_CATALOG", SQL_IDENTIFIER);
            addColumn(t, "SEQUENCE_SCHEMA", SQL_IDENTIFIER);
            addColumn(t, "SEQUENCE_NAME", SQL_IDENTIFIER);
            addColumn(t, "DATA_TYPE", CHARACTER_DATA);
            addColumn(t, "NUMERIC_PRECISION", CARDINAL_NUMBER);
            addColumn(t, "NUMERIC_PRECISION_RADIX", CARDINAL_NUMBER);
            addColumn(t, "NUMERIC_SCALE", CARDINAL_NUMBER);
            addColumn(t, "MAXIMUM_VALUE", CHARACTER_DATA);
            addColumn(t, "MINIMUM_VALUE", CHARACTER_DATA);
            addColumn(t, "INCREMENT", CHARACTER_DATA);
            addColumn(t, "CYCLE_OPTION", YES_OR_NO);
            addColumn(t, "DECLARED_DATA_TYPE", CHARACTER_DATA);
            addColumn(t, "DECLARED_NUMERIC_PRECISION", CARDINAL_NUMBER);
            addColumn(t, "DECLARED_NUMERIC_SCALE", CARDINAL_NUMBER);

            // HSQLDB-specific
            addColumn(t, "START_WITH", CHARACTER_DATA);
            addColumn(t, "NEXT_VALUE", CHARACTER_DATA);

            // order SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME
            // false PK, as CATALOG may be null
            HsqlName name = HsqlNameManager.newInfoSchemaObjectName(
                sysTableHsqlNames[SYSTEM_SEQUENCES].name, false,
```

```
32                  SchemaObject.INDEX);

34              t.createPrimaryKeyConstraint(name, new int[] {
                    0, 1, 2
36              }, false);

38              return t;
            }
40
            //
42          final int sequence_catalog           = 0;
            final int sequence_schema            = 1;
44          final int sequence_name              = 2;
            final int data_type                  = 3;
46          final int numeric_precision          = 4;
            final int numeric_precision_radix    = 5;
48          final int numeric_scale              = 6;
            final int maximum_value              = 7;
50          final int minimum_value              = 8;
            final int increment                  = 9;
52          final int cycle_option               = 10;
            final int declared_data_type         = 11;
54          final int declared_numeric_precision = 12;
            final int declared_numeric_scale     = 13;
56          final int start_with                 = 14;
            final int next_value                 = 15;
58
            //
60          Iterator       it;
            Object[]       row;
62          NumberSequence sequence;

64          it = database.schemaManager.databaseObjectIterator(
                SchemaObject.SEQUENCE);
66
            while (it.hasNext()) {
68              sequence = (NumberSequence) it.next();

70              if (!session.getGrantee().isAccessible(sequence)) {
                    continue;
72              }

74              row = t.getEmptyRowData();

76              NumberType type = (NumberType) sequence.getDataType();
                int radix =
78                  (type.typeCode == Types.SQL_NUMERIC || type.typeCode == Types
                        .SQL_DECIMAL) ? 10
80                                    : 2;

82              row[sequence_catalog] = database.getCatalogName().name;
                row[sequence_schema]  = sequence.getSchemaName().name;
84              row[sequence_name]    = sequence.getName().name;
                row[data_type]        = sequence.getDataType().getFullNameString();
```

```
86              row[numeric_precision] =
                    ValuePool.getInt((int) type.getPrecision());
88              row[numeric_precision_radix]    = ValuePool.getInt(radix);
                row[numeric_scale]              = ValuePool.INTEGER_0;
90              row[maximum_value] = String.valueOf(sequence.getMaxValue());
                row[minimum_value] = String.valueOf(sequence.getMinValue());
92              row[increment] = String.valueOf(sequence.getIncrement());
                row[cycle_option]               = sequence.isCycle() ? "YES"
94                                                                   : "NO";
                row[declared_data_type]         = row[data_type];
96              row[declared_numeric_precision] = row[numeric_precision];
                row[declared_numeric_scale]     = row[declared_numeric_scale];
98              row[start_with] = String.valueOf(sequence.getStartValue());
                row[next_value]                 = String.valueOf(sequence.peek());
100
                t.insertSys(session, store, row);
102         }

104         return t;
        }
```

Biggest clone (82 SLOC)

```
1   // Location /hsqldb/src/org/hsqldb/dbinfo/DatabaseInformationFull.java (2687,2746)
    // Other locations:
3   // /hsqldb/src/org/hsqldb/dbinfo/DatabaseInformationFull.java (5967, 6026)
    // /hsqldb/src/org/hsqldb/dbinfo/DatabaseInformationFull.java (3440, 3495)
5   // /hsqldb/src/org/hsqldb/dbinfo/DatabaseInformationFull.java (7932, 7985)
    // common type block
7               if (type.isCharacterType()) {
                    row[character_maximum_length] =
9                       ValuePool.getLong(type.precision);
                    row[character_octet_length] =
11                      ValuePool.getLong(type.precision * 2);
                    row[character_set_catalog] =
13                      database.getCatalogName().name;
                    row[character_set_schema] =
15                      ((CharacterType) type).getCharacterSet()
                            .getSchemaName().name;
17                  row[character_set_name] =
                        ((CharacterType) type).getCharacterSet().getName()
19                          .name;
                    row[collation_catalog] = database.getCatalogName().name;
21                  row[collation_schema] =
                        ((CharacterType) type).getCollation().getSchemaName()
23                          .name;
                    row[collation_name] =
25                      ((CharacterType) type).getCollation().getName().name;
                } else if (type.isNumberType()) {
27                  row[numeric_precision] = ValuePool.getLong(
                        ((NumberType) type).getNumericPrecisionInRadix());
29                  row[declared_numeric_precision] = ValuePool.getLong(
                        ((NumberType) type).getNumericPrecisionInRadix());
31
```

```
                    if (type.isExactNumberType()) {
                        row[numeric_scale] = row[declared_numeric_scale] =
                            ValuePool.getLong(type.scale);
                    }

                    row[numeric_precision_radix] =
                        ValuePool.getLong(type.getPrecisionRadix());
                } else if (type.isBooleanType()) {

                    //
                } else if (type.isDateTimeType()) {
                    row[datetime_precision] = ValuePool.getLong(type.scale);
                } else if (type.isIntervalType()) {
                    row[data_type] = "INTERVAL";
                    row[interval_type] =
                        ((IntervalType) type).getQualifier(type.typeCode);
                    row[interval_precision] =
                        ValuePool.getLong(type.precision);
                    row[datetime_precision] = ValuePool.getLong(type.scale);
                } else if (type.isBinaryType()) {
                    row[character_maximum_length] =
                        ValuePool.getLong(type.precision);
                    row[character_octet_length] =
                        ValuePool.getLong(type.precision);
                } else if (type.isBitType()) {
                    row[character_maximum_length] =
                        ValuePool.getLong(type.precision);
                    row[character_octet_length] =
                        ValuePool.getLong(type.precision);
                } else if (type.isArrayType()) {
                    row[maximum_cardinality] =
                        ValuePool.getLong(type.arrayLimitCardinality());
                    row[data_type] = "ARRAY";
                }
```

Biggest clone class by total SLOC example clone (212 total SLOC)

```
// Location /hsqldb/src/org/hsqldb/test/TestAcl.java (310,319)
{
            acl = (ServerAcl) aclPermitLocalhosts[i];

            assertTrue(
                "Denying access from localhost with localhost-permit ACL",
                acl.permitAccess(localhostByAddr.getAddress()));
            assertFalse(
                "Permitting access from other host with localhost-permit ACL",
                acl.permitAccess(otherHostByAddr.getAddress()));
        }
```

Example 1/5

```
// /hsqldb/src/org/hsqldb/test/TestAcl.java (283,290)
{
            acl = (ServerAcl) aclDenyAlls[i];
```

```
            assertFalse("Permitting access from localhost with deny-all ACL",
                        acl.permitAccess(localhostByAddr.getAddress()));
            assertFalse("Permitting access from other host with deny-all ACL",
                        acl.permitAccess(otherHostByAddr.getAddress()));
        }
```

Example 2/5

```
// /hsqldb/src/org/hsqldb/test/TestAcl.java (339,348)
{
            acl = (ServerAcl) aclDenyLocalhosts[i];

            assertFalse(
                "Permitting access from localhost with localhost-deny ACL",
                acl.permitAccess(localhostByAddr.getAddress()));
            assertTrue(
                "Denying access from other host with localhost-deny ACL",
                acl.permitAccess(otherHostByAddr.getAddress()));
        }
```

Example 3/5

```
// /hsqldb/src/org/hsqldb/test/TestAcl.java (367,375)
{
            acl = (ServerAcl) aclPermitLocalNets[i];

            assertTrue("Denying access from localNet with localNet-permit ACL",
                       acl.permitAccess(localhostByAddr.getAddress()));
            assertFalse(
                "Permitting access from other Net with localNet-permit ACL",
                acl.permitAccess(otherHostByAddr.getAddress()));
        }
```

Example 4/5

```
// /hsqldb/src/org/hsqldb/test/TestAcl.java (395,403)
{
            acl = (ServerAcl) aclDenyLocalNets[i];

            assertFalse(
                "Permitting access from localNet with localNet-deny ACL",
                acl.permitAccess(localhostByAddr.getAddress()));
            assertTrue("Denying access from other Net with localNet-deny ACL",
                       acl.permitAccess(otherHostByAddr.getAddress()));
        }
```

Example 5/5

References

[Bax+98] Ira D Baxter et al. "Clone detection using abstract syntax trees". In: *Software Maintenance, 1998. Proceedings., International Conference on*. IEEE. 1998, pp. 368–377.

[Bel+07] S. Bellon et al. "Comparison and Evaluation of Clone Detection Tools". In: *IEEE Transactions on Software Engineering* 33.9 (Sept. 2007), pp. 577–591. ISSN: 0098-5589. DOI: 10.1109/TSE.2007.70725.

[Clo18] CloneDR. *CloneDR duplication documentation*. 2018. URL: http://www.semdesigns.com (visited on 12/14/2018).

[Cod18] Codeclimate. *Codeclimate duplication documentation*. 2018. URL: https://docs.codeclimate.com/docs/duplication-concept (visited on 12/14/2018).

[HBJ12] Benedikt Hauptmann, Veronika Bauer, and Maximillian Junker. "Using edge bundle views for clone visualization". In: (June 2012). DOI: 10.1109/IWSC.2012.6227877.

[Hol06] D. Holten. "Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data". In: *IEEE Transactions on Visualization and Computer Graphics* 12.5 (Sept. 2006), pp. 741–748. ISSN: 1077-2626. DOI: 10.1109/TVCG.2006.147.

[Kos08] Rainer Koschke. "Identifying and removing software clones". In: *Software Evolution*. Springer, 2008, pp. 15–36.

[MF] M Mohammed and J Fawcett. "Clone Detection Using Scope Trees". In: ().

[RC08] Chanchal K. Roy and James R. Cordy. "Scenario-Based Comparison of Clone Detection Techniques". In: *Proceedings of the 2008 The 16th IEEE International Conference on Program Comprehension*. ICPC '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 153–162. ISBN: 978-0-7695-3176-2. DOI: 10.1109/ICPC.2008.42. URL: https://doi.org/10.1109/ICPC.2008.42.

[SFM99] M-AD Storey, F David Fracchia, and Hausi A Müller. "Cognitive design elements to support the construction of a mental model during software exploration". In: *Journal of Systems and Software* 44.3 (1999), pp. 171–185.

[SR15] Jeffrey Svajlenko and Chanchal K. Roy. "Evaluating Clone Detection Tools with BigCloneBench". In: *Proceedings of the 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. ICSME '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 131–140. ISBN: 978-1-4673-7532-0. DOI: 10.1109/ICSM.2015.7332459. URL: http://dx.doi.org/10.1109/ICSM.2015.7332459.

[SS15] Mythili ShanmughaSundaram and Sarala Subramani. "A measurement of similarity to identify identical code clones". In: *Int Arab J Inform Technol* 12 (2015), pp. 735–740.