An Application Tool for Visualizing Research Work on Landslides

Katarina Lepp
Abstract

An Application Tool for Visualizing Research Work on Landslides

Katarina Lepp

This report describes the process of organizing the research material of a PhD thesis into a database, and the development of an application in order to access the information. The thesis relates to rainfall-induced landslides in the capital of Honduras: The data are a collection of press-based information related to these landslides over a period of 26 years and stored in several Excel files. The task has been to analyze the data and organize them into a conceptual database model. After processing, the data were transferred from CSV files into a SQLite database. A friendly and intuitive graphical user interface (GUI) was developed to allow users to query the database according to the specifications made by the author of the thesis. The queries of the application together with the database tables were tuned for performance according to database design principles and the SQLite documentation.
# Table of Contents

Table of Contents ........................................................................................................................................... 0

1. Introduction .................................................................................................................................................. 1

2. Background .................................................................................................................................................. 2
   2.1 Database design concepts ..................................................................................................................... 2
   2.2 Comparison of the optimization process in SQLite and Oracle Database ........................................... 3

3. Design ....................................................................................................................................................... 5
   3.1 Original organization of the landslides data .......................................................................................... 5
   3.2 Normalization ......................................................................................................................................... 7
       3.2.1 Normalization of the landslides table ............................................................................................. 7
       3.2.2 Normalization of the rainfall table ................................................................................................. 10

4. Implementation ......................................................................................................................................... 11
   4.1 Population of the database .................................................................................................................... 11
   4.2 Queries in the application ....................................................................................................................... 12
       4.2.1 Queries relating to landslide data ................................................................................................. 12
       4.2.2 Queries relating to rainfall data .................................................................................................... 15
   4.3 Development of the Graphical User Interface ....................................................................................... 16
       4.3.1 The landslide search tab ............................................................................................................... 17
       4.3.2 The map search tab ...................................................................................................................... 20
       4.3.2 The Rainfall search tab ............................................................................................................... 21

5. Performance tuning .................................................................................................................................... 23
   5.1 Text search ............................................................................................................................................. 23
   5.2 Creating indices ...................................................................................................................................... 23
   5.3 Prepared queries .................................................................................................................................... 24

6. Measurement Results ............................................................................................................................... 25

7. Related work .............................................................................................................................................. 27

8. Conclusions and future work .................................................................................................................... 29

References ....................................................................................................................................................... 30

Appendix I. Create statements for the tables ................................................................................................. 31

Appendix II. Contents of the Excel files ........................................................................................................ 34

Appendix III. Excel tables ............................................................................................................................. 38
    The Landslide Excel table ........................................................................................................................ 38
    The Rainfall Excel table ........................................................................................................................... 42
1. Introduction

In the capital of Honduras, Tegucigalpa, landslides occur on a yearly basis. During the rainy season, heavy rainfall weakens the ground and gives rise to avalanches of mud, debris and rocks capable of causing injuries and casualties, destroying homes and damaging other infrastructure. Despite the dangers of living on steep slopes and riverbanks, much of Tegucigalpa's development in recent years has taken place in these risky areas. Although the main cause of the landslides is rainfall, human activities have also contributed to the occurrence of these destructive events [2].

The PhD student Elias Garcia-Urquia at the Department of Applied Mechanics at Uppsala University is conducting research about the correlation between the amount of rainfall and the occurrence of landslides. The thesis uses a compilation of data relating landslides and rainfall during the period 1980 to 2005. As the information in the database is stored in Excel files, its access is difficult through traditional search. In addition, there is the need for conducting more complicated searches, such as knowing which dates had a cumulative rainfall above a certain value, or which landslides that were caused by broken pipes.

The data about the nearly 400 landslides compiled in the database concern the date of occurrence, the location, the type of movement according to well-established landslide classifications and the damages caused. Information about the sources of the data is also available for further references; these are mainly the two dominating newspapers in Tegucigalpa, Honduras, La Tribuna and El Heraldo. The database also stores data about the amount of daily and antecedent rainfall that has contributed to the initiation of the landslides.

The aim of this report is to describe the process of organizing the data relating to the landslides into a database, and developing an application for accessing it.
2. Background

The first sub-section of this chapter introduces the database concepts that are used through this report. The second sub-section contains a comparison between the optimization process in SQLite and another RDBMS (Relational Database Management System); the Oracle Database.

2.1 Database design concepts

The database concepts used through the thesis report are introduced as follows:

**Table**

A table, also called relation, is a collection of data relating a real life entity. For example a table could be called Landslide and contain data about landslides. Each column in the table represents an attribute for that entity, for instance location, and each row in the table corresponds to one particular landslide [1].

**Index**

An index is a data structure that the database manager can store to speed up the writes and reads to the database. The index is created for one column or several columns in a table. When using an index SQLite does a binary search in the index table to find the first match for the searched for value, takes the corresponding row id and does a binary search on the table to get the result. If the index covers all the columns that should be returned, the binary search in the table is unnecessary. Such an index is called a covering index in SQLite [1].

**SQL**

“SQL” stands for Structured Query Language and is a database manipulation language that is the standard language for commercial relational database management systems [1].

**SQLite**

SQLite is an embedded relational database. It was designed to minimize the overhead that other Relational Database Manager Systems (RDBMS) often have. SQLite supports most of the SQL standard. It is open source. Some features that are not implemented from SQL are stored procedures [7].

**Database normalization**

As stated by Elmasri [1] this is a summary of the definition of database normalization
Four guidelines to a good relational database design can be described as follows:

1. The semantics of the attributes should be clear
2. The design should reduce the redundant information in tuples
3. The design should reduce null values
4. Avoid the possibility of generation of spurious tuples, that is, ensure good mapping of primary keys to foreign keys

The process of normalization minimizes redundancy and insertion, deletion and update-anomalies. The state of a table can be measured by what normal form it is in. To explain the definitions of the different normal forms there is a need to explain the concept of functional dependency.

A functional dependency is a constraint between two subsets of attributes among the attributes of a relation. One subset is said to determine the other subset if two rows with the same values for the attributes in the first subset means that they will have the same values for the attributes in the other subset. In a relation 'Person', that contains name and social security number (ssn) for a person, the ssn will determine the name since the ssn uniquely defines a person. The name attribute does not determine the ssn because there are persons that share the same name. A primary key is a subset of attributes that determines all other attributes not included in the key. A candidate key is a primary key that contains no unnecessary attributes. That means that no attribute can be removed from the candidate key so that it would still determine all other attributes.

There are many normal forms, but for this purpose the interest is in achieving the third normal form or Boyce-Codd Normal Form. The first normal form states that the values of the attributes have to be atomic. The second normal form requires that no dependencies of a part of a candidate key should determine a non-key attribute. The third normal form states that no transitive relations should exists, that is dependencies on the form $X \rightarrow Y \rightarrow Z$. The next normal form is called Boyce-Codd Normal Form. (BCNF) It is a higher normal form than third normal form and is not always possible to achieve. It states that all non-key attributes have to be dependent on a candidate key and that the relation is in first normal form.

### 2.2 Comparison of the optimization process in SQLite and Oracle Database

In this section the query optimization techniques of SQLite is compared with one of the most used RDBMS's, i.e. the Oracle Database. The information in this chapter is derived from the Oracle Database SQL Tuning guide [3] and SQLite's official web page [7].

The query optimization is important from a performance perspective, and the way SQLite handles it differs from typical client-server architecture database manager systems. SQLite does not have a separate server process but it exists within the same process that hosts it. The
Oracle Database is one of the dominating database manager software today, so it represents a good reference point when examining query optimization in SQLite.

Oracle uses Structured Query Language (SQL), which SQLite also implements. SQL is a non-procedural language, which means that the optimizer can reorganize, merge and process the order of execution. The task of the database optimizer is to evaluate the cost of the different possible ways of executing statements in SQL, and to choose the one with the lowest cost. Both Oracle and SQLite use this method of optimizing, which is called cost-based optimization. The cost is calculated by usage of system resources, which includes I/O, CPU and memory, the number of rows returned and the size of the data sets.

In both SQLite and Oracle the query can be replaced by a more optimal but semantically identical query. SQLite has predefined ways of operating the different kinds of queries, the main optimizing are for the joining of tables. The Oracle optimizer can choose between different join methods, whereas SQLite always uses nested loops join.

The optimizing in SQLite goes in the following way:
1. The optimizer chooses the nested order of the loops, and then chooses good indices for each loop.
2. The loop order is calculated by making a directed weighted graph and finding the minimum cost path through it.
3. The optimizer then makes the decision of which indices to use, or decides to do a full table scan.

The Oracle optimizer can choose between different access methods for querying, such as full scan or index scan. Oracle uses statistics to optimize the queries to a great degree as opposed to SQLite. For example, if searching for entries that have a certain attribute, Oracle will use a full table scan if the majority of the rows have this value, otherwise it will use the index scan and fetch them with row id (an index assigned to every row by the DBMS, invisible to the user). The statistics gathering in SQLite can be increased with the statement Analyze. When Analyze is run, it creates a table with information about the distribution of keys within an index.

When there is not enough statistics to create an execution plan, the Oracle optimizer uses an adaptive plan where the final decision varies during the execution time. The adaptive plan contains multiple predetermined sub plans, and an optimizer statistics collector. In a case that the optimizer makes a miss-estimate, the optimizer will notice that and choose a better plan next time. This is quite a different strategy from that in SQLite. The SQLite optimizer will always execute the same query plan for a given SQL statement. This is intentional, since SQLite is an embedded database and has to behave in a predictable way.

To summarize, SQLite executes faster than most of the other DBMS’s for simple queries, such as a single insert statement, due to the lesser amount of overhead. When the queries become more complex DBMS’s with better optimizers perform better.
3. Design

This chapter describes the process of designing the relational database model for the data in the Excel files.

The landslides related data were stored in one Excel sheet, and the rainfall related data in one sheet. The landslides Excel sheet contains the largest amount of data with about 50 columns; while the rainfall Excel sheet contains only three columns of importance.

3.1 Original organization of the landslides data

The first step towards creating a database model was to analyze the content of the Excel tables. 

*To view the Excel sheets see the Appendix III.*

![Figure 1. A part of the landslides Excel sheet](image)

Fig. 1 shows what the original landslides Excel table looked like. The approximately 50 columns contain facts about landslides, their newspaper sources, and the reliability of the publishing houses that published the newspapers. The landslides are assigned an id number that is unique for each landslide.

The basic attributes for a landslide in the scope of this database are the time and date, the location, the type and the cause, the damage it caused, and also the newspaper articles that mention the landslide. Each landslide has data about the reliability of the publishing houses, which mostly are the two main publishing houses in Honduras: La Tribuna and El Heraldo.

Besides the relevant data the Excel sheets contains columns that can be removed, because they contain redundant information. They are the ones that contain data that are already stored...
in another column, or are computed from the data in other columns. For example the column
MONTH_NUMBER (see Fig. 1) is computed as the number of months between the date and the
beginning of the year 1980.

Besides MONTH_NUMBER, the columns with redundant data are:
  • The DATE or SOURCE columns derive the MONTH and YEAR columns.
  • The SINGLE VRS. MULTIPLE column can be computed by checking if any other
    landslide occurred within seven days in the same location.
  • RAINFALL INTENSITY and DAILY RAINFALL TONCONTIN are stored in other tables.
  • LT, EH and TOTAL are derived from the reliability of publishing house columns.
  • The damage related columns (DEATHS, INJURED etc.) are derived from the string in
    ADDITIONAL RELEVANT INFORMATION column, which is also the case for the
    DURATION OF RAINFALL and START OF RAINFALL columns.

![Figure 2. A part of the rainfall Excel sheet](image)

Fig. 2 shows the rainfall table. It contains the date (DATE), the amount of rainfall for that date
(TRIGG. RAINFALL) and the number of slides that occurred in that date (SLIDES). The rest of
the columns are derived from other columns like the DAY, DAY, MONTH and YEAR columns,
which are derived from DATE, or else they were added for the convenience of Garcia-Urquia.

To summarize, the following columns are removed from the landslides Excel table:
  • MONTH_NUMBER
  • MONTH
  • YEAR
  • SINGLE VRS. MULTIPLE
  • RAINFALL INTENSITY
  • DAILY RAINFALL TONCONTIN
  • LT
The columns with values that are derived from the ADDITIONAL RELEVANT INFORMATION however are kept in the table. The alternative would be to only keep the ADDITIONAL RELEVANT INFORMATION column and then extract the data for the nine categories of damage (number of deaths, injured etc.) from the string values in it. This was considered to be too time consuming because of the ambiguity of the data. In the rainfall table all columns are removed except the DATE, TRIGG. RAINFALL and SLIDES columns.

The database model will contain one table for the landslides data and one table for the rainfall data. The primary keys for the landslides table will be ID, and the primary key for the rainfall table will be DATE, since date is unique for every day. To reduce redundancy and update anomalies the tables are normalized to third normal form, or Boyce-Codd Normal Form (BCNF) when possible.

### 3.2 Normalization

After understanding the contents of the Excel tables, the next step was to normalize the tables in the database model in order to minimize redundancy and the risk of modification anomalies.

#### 3.2.1 Normalization of the landslides table

The landslides table is not in first normal form since it contains non-atomic values in the SOURCE and TYPE OF SLIDE columns see Fig. 3. Both SOURCE and TYPE OF SLIDE are composite and multivalued attributes. The SOURCE attribute is composed by the name of a newspaper and the date the article was published. Movement and material compose the TYPE OF SLIDE attribute.

To meet the conditions of the first Normal Form the columns with non-atomic values have to be transformed into columns with only atomic values. The result of this transformation is shown in Fig. 4. Then, according to the rule for handling multivalued attributes, the decomposed SOURCE and TYPE OF SLIDE columns are moved to new tables along with the primary key ID from the original table. The primary keys for the new tables are \{id, name, date\} for the SOURCE table which is renamed Source and \{id, movement, material\} for the TYPE OF SLIDE table which is renamed Type. The new tables are shown in Fig. 5.
A problem with the landslides Excel table is that it stores the same reliability of publishing house columns for La Tribuna and El Heraldo, as seen in Fig. 6 and Fig. 7. This design causes redundancy since there are not always articles from both sources for each landslide.
Furthermore there might be other publishing houses than La Tribuna and El Heraldo, which would be impossible to store with the current mode of storing. There are indeed articles from other newspapers, for example La Prensa. To reduce redundancy and to make it possible to store data for any publishing house a new table Reliability_of_Publishing_House was created with the landslides ID attribute and name of the publishing house as primary key (see Fig. 8).

<table>
<thead>
<tr>
<th>WHEN 0.25</th>
<th>DAMAGES 0.25</th>
<th>CAUSES 0.1</th>
<th>TYPE 0.1</th>
<th>IMAGES Y/N</th>
<th>FOLLOW-UP Y/N</th>
<th>EXCLUSIVE Y/N</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.25</td>
<td>0.1</td>
<td>O</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0.125</td>
<td>0</td>
<td>O</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>0</td>
<td>0.125</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>0</td>
<td>0.25</td>
<td>0.1</td>
<td>O</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>0</td>
<td>0.25</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>0</td>
<td>0.25</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>0.25</td>
<td>0.25</td>
<td>0.1</td>
<td>O</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>0.25</td>
<td>0.25</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>0.25</td>
<td>0.25</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>0.25</td>
<td>0.25</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>0.25</td>
<td>0.25</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

*Figure 6. The reliability of publishing house columns for La Tribuna*

<table>
<thead>
<tr>
<th>WHEN 0.25</th>
<th>DAMAGES 0.25</th>
<th>CAUSES 0.1</th>
<th>TYPE 0.1</th>
<th>IMAGES Y/N</th>
<th>FOLLOW-UP Y/N</th>
<th>EXCLUSIVE Y/N</th>
<th>LT</th>
<th>EH</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.25</td>
<td>0.1</td>
<td>O</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>0.9</td>
<td>0.9</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
<td></td>
<td>0.425</td>
<td>0</td>
<td>1,725</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
<td></td>
<td>0.425</td>
<td>0</td>
<td>1,725</td>
</tr>
<tr>
<td>0.25</td>
<td>0.25</td>
<td>0.1</td>
<td>O</td>
<td>Y</td>
<td>N</td>
<td></td>
<td>0.9</td>
<td>0.9</td>
<td>3.6</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
<td></td>
<td>0.65</td>
<td>0</td>
<td>1.3</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
<td></td>
<td>0.525</td>
<td>0</td>
<td>1.05</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
<td></td>
<td>0.65</td>
<td>0</td>
<td>1.3</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
<td></td>
<td>0.65</td>
<td>0</td>
<td>1.3</td>
</tr>
<tr>
<td>0.25</td>
<td>0.25</td>
<td>0.1</td>
<td>O</td>
<td>Y</td>
<td>N</td>
<td></td>
<td>0.9</td>
<td>0.9</td>
<td>1.8</td>
</tr>
<tr>
<td>0.25</td>
<td>0.25</td>
<td>0.1</td>
<td>O</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>0.9</td>
<td>0.9</td>
<td>2.7</td>
</tr>
<tr>
<td>0.25</td>
<td>0.25</td>
<td>0.1</td>
<td>O</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>0.65</td>
<td>0.65</td>
<td>1.95</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
<td></td>
<td>0.65</td>
<td>0</td>
<td>1.95</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
<td></td>
<td>0.65</td>
<td>0</td>
<td>1.95</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
<td></td>
<td>0</td>
<td>0.65</td>
<td>6.2</td>
</tr>
<tr>
<td>0.25</td>
<td>0.25</td>
<td>0.1</td>
<td>O</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>0.9</td>
<td>0.9</td>
<td>2.55</td>
</tr>
<tr>
<td>0.25</td>
<td>0.25</td>
<td>0.1</td>
<td>O</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>0</td>
<td>0.5</td>
<td>9.5</td>
</tr>
<tr>
<td>0.25</td>
<td>0.25</td>
<td>0.1</td>
<td>O</td>
<td>Y</td>
<td>N</td>
<td></td>
<td>0.9</td>
<td>0.9</td>
<td>7.2</td>
</tr>
<tr>
<td>0.25</td>
<td>0.25</td>
<td>0.1</td>
<td>O</td>
<td>Y</td>
<td>N</td>
<td></td>
<td>0.9</td>
<td>0.5</td>
<td>2.7</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>N</td>
<td>N</td>
<td></td>
<td>0</td>
<td>0.3</td>
<td>0.3</td>
</tr>
</tbody>
</table>

*Figure 7. The reliability of publishing house columns for El Heraldo*
The attributes that remain after this division are stored in a table Landslide with ID as primary key. The names of the attributes are changed to lower case so for example ID becomes 'id'. The tables in the new design are shown in Fig. 9. In Landslide the only functional dependency is that id determines all other attributes. The table is in BCNF since it is in first normal form and all non-key attributes are dependent on a candidate key. The Type and Source tables are in BCNF for the same reasons. The name and id in Reliability_of_Publishing_House represents a candidate key and determines all other attributes. There are no other dependencies and so the table is in BCNF.

3.2.2 Normalization of the rainfall table

In the rainfall table the only determinant is date, and it determines triggering_rainfall and slides. It is in BCNF and no normalization is needed. It is shown in Fig. 9.
4 Implementation

The language used for implementing the application is Java. To avoid any need for installation and administration of a database management system (DBMS) the application uses an embedded database management system: SQLite. This makes the application portable and requires no knowledge of database administration from the end user.

This chapter describes the process of populating the database, implementing the queries that the application uses and implementing the user interface.

4.1 Population of the database

To populate the designed database tables the Excel documents were converted into CSV (Comma Separated Values) files. The contents of a CSV file can be loaded into a table via the import command in the SQLite command line utility. The full landslides Excel tables were loaded into a large bulk table and the other tables were populated by selecting columns from the bulk table and inserting them. The Rainfall table could be populated directly since it only contained two columns by loading the CSV file into the table.

An issue with importing CSV files is that SQLite fails to recognize empty strings as null. So for every column that might contain null values the empty strings were changed to null. The triggering_rainfall had a similar problem were some values were 'NaN'. These values were set to 0.0.

Some of the values of the columns in the landslides Excel table had to be parsed before they could be stored. This was the case for the SOURCE and TYPE OF SLIDE columns (see Table 1). The parser was written in Java. An example of the Source value is "La Tribuna, El Heraldo (07,08-jun-05)" which was parsed as "La Tribuna (07-jun-05)", "El Heraldo, (07-jun-05)", "La Tribuna (08-jun-05)", "El Heraldo, (08-jun-05)".

The type values were divided into different categories of movement according to the Varnes’ classification: Slide, Falls, Spreads, Topples, Flows [8]. The material or materials for each slide was stored as a string. The atomic strings were put in CSV files that were imported into the Source and Type -tables respectively.
Table 1. Part of the TYPE OF SLIDE and SOURCE columns from the landslides Excel table

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>TYPE OF SLIDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>La Tribuna [30-Jul-96]</td>
<td>Rockfall and debris flow</td>
</tr>
<tr>
<td>El Heraldo [30-Jul-96]</td>
<td>Rockfall</td>
</tr>
<tr>
<td>La Tribuna [21-Aug-96]</td>
<td>Rockfall</td>
</tr>
<tr>
<td>El Heraldo [26-Aug-96]</td>
<td>Rockfall</td>
</tr>
<tr>
<td>La Tribuna [09-Oct-96]</td>
<td></td>
</tr>
<tr>
<td>La Tribuna [09-Oct-96]</td>
<td></td>
</tr>
<tr>
<td>La Tribuna [09-Oct-96]</td>
<td></td>
</tr>
<tr>
<td>La Tribuna [09-Oct-96]</td>
<td></td>
</tr>
<tr>
<td>El Heraldo [10-Oct-96]</td>
<td></td>
</tr>
<tr>
<td>La Tribuna [17, 18-Oct-96]</td>
<td>Rockfall</td>
</tr>
<tr>
<td>La Tribuna [17, 18-Oct-96], El Heraldo [18-Oct-96]</td>
<td></td>
</tr>
</tbody>
</table>

4.2 Queries in the application

This chapter describes the queries specified by Elias Garcia-Urquia that the application implements. The queries can be distinguished into two categories: queries relating to landslides data and queries relating to rainfall data.

4.2.1 Queries relating to landslide data

The landslides data can be searched by:

- Location; which is the place where the landslide occurred. The user is able to search not only the name of the neighborhoods but also other location related names, for example search for ‘Hill’ or ‘school’ or the name of a place.
- Time interval; the user can search for landslides that occurred during a year, a month, or on a specific date. It is also possible to search between two points in time that is between two years or two dates. Then, the search result will include the start point and the end point. If the user inputs two months the result will be the landslides that happened between those months for all the years recorded in the database.
- Negative impact; called level of damage in the application. It is a scale to measure the impact of a landslide.
- Keywords; the user can search on type, cause and damage to public infrastructure.
- Time of the day, which can be an exact time, or a time in the day, or null.

The result of a user search for landslides returns the date, location, cause, type (movement and material), time of the day and the level of damage for each landslide matching the search
conditions. The user can view the sources where a landslide has been described. The search conditions can be combined.

Below example SQL queries for searching the landslides data are presented. When using text search in SQLite it is not possible to join a normal table and a text search table and then query columns in both text search table and normal table. This is solved in the queries by taking the union of queries, as seen in the query for keywords. Since all of the search conditions can be combined, the result of the queries that involve text searching has to be intersected with the normal query when the text search options are combined with the other options. The queries that uses text search are the query on location and the query that searches for a keyword. The final query looks as follows:

```
Query for date, time etc.
INTERSECT
Query for location
INTERSECT
Query for keyword
```

**Search for landslides data at some location**
Returns landslides data for landslides where the location(s) matches the search word.

```
SELECT Landslide.id as id, date, time, location, movement, material, cause, level_of_damage
FROM ((SELECT id as ts_id FROM Text_search WHERE location MATCH 'Reparto')
JOIN Landslide ON ts_id = Landslide.id LEFT OUTER JOIN Type ON Landslide.id = Type.id)
```

**Search for landslides data at some date or a time interval**
Depending on what search field the user writes in the following query will be generated: For a specific date; retrieve landslides data for landslides where date is equal to the search word. When searching for the neighborhood Reparto the query becomes:

```
SELECT Landslide.id as id, date, time, location, movement, material, cause, level_of_damage
FROM Landslide LEFT OUTER JOIN Type ON Landslide.id = Type.id
WHERE date = '1992-06-01' OR earliest_date_reported_date = '1992-06-01'
```

For a year and a month; retrieve landslides data for landslides where dates are within a range. For example for year 1992 and month 6 (June) the query becomes:

```
SELECT Landslide.id as id, date, time, location, movement, material, cause, level_of_damage
FROM Landslide LEFT OUTER JOIN Type ON Landslide.id = Type.id
```
For a month; retrieve landslides data for landslides where dates contain the month string.
For example for month 6 (June) the query becomes:

```sql
SELECT Landslide.id as id, date, time, location, movement, material, cause, level_of_damage
FROM Landslide  LEFT OUTER JOIN Type ON Landslide.id = Type.id
WHERE date like "-%06-" OR earliest_date_reported like "-%06-
```

**Search for landslides data at some time**
There are four options to search for landslides with a condition time of the day. The default choice is to have no restrictions on the time of day. If the exact clock time is known a user can pose a query as; Retrieve all data where the time is not specified by 'evening', 'morning' but it is not unknown, i.e. it is not null. This will return data for landslides that occurred at exact clock times.

```sql
SELECT Landslide.id as id, date, time, location, movement, material, cause, level_of_damage
FROM Landslide LEFT OUTER JOIN Type ON Landslide.id = Type.id
WHERE time != 'early morning' AND time != 'morning' AND time != 'afternoon' AND time != 'evening' AND time != 'midnight'
```

If the time of the day is known it can be defined a query as; Retrieve all landslides data where the time is specified by 'evening', 'morning' etc.

```sql
SELECT Landslide.id as id, date, time, location, movement, material, cause, level_of_damage
FROM Landslide  LEFT OUTER JOIN Type ON Landslide.id = Type.id
WHERE  time = 'early morning' OR time = 'morning' OR time = 'afternoon' OR time = 'evening' OR time = 'midnight'
```

If the time is unknown return all rows where time is null. Query:

```sql
SELECT Landslide.id as id, date, time, location, movement, material, cause, level_of_damage
FROM Landslide  LEFT OUTER JOIN Type ON Landslide.id = Type.id
WHERE time IS NULL
```

**Search for landslides data with a determined level of damage**
Returns landslides data for landslides caused a specific damage level, defined by a number. Querying for landslides with level of damage equal to 5:

```sql
SELECT Landslide.id as id, date, time, location, movement, material, cause, level_of_damage
FROM Landslide  LEFT OUTER JOIN Type ON Landslide.id = Type.id
WHERE level_of_damage = 5
```
Search for landslides data by cause, type and damage to public infrastructure
Returns landslides data for landslides where the landslide has a cause, a type (material or movement) or damage to public infrastructure that match or contains a keyword. When searching for landslides related to 'school' the query becomes:

```
SELECT Landslide.id as id, date, time, location, movement, material, cause
level_of_damage
FROM Landslide  LEFT OUTER JOIN Type ON Landslide.id = Type.id
WHERE movement = 'school' OR material = ' school'
UNION ALL
SELECT Landslide.id as id, date, time, location, movement, material, cause
level_of_damage
FROM ((SELECT id AS ts_id FROM Text_search WHERE cause MATCH 'school')
JOIN Landslide ON ts_id = Landslide.id LEFT OUTER JOIN Type ON Landslide.id = Type.id)
UNION ALL
SELECT Landslide.id as id, date, time, location, movement, material, cause,
level_of_damage
FROM ((SELECT id AS ts_id FROM Text_search where damage_to_public_infrastructure
MATCH 'school') JOIN Landslide ON ts_id = Landslide.id LEFT OUTER JOIN Type ON
Landslide.id = Type.id)
```

Search for landslide sources
Returns all sources that mention the landslide with a certain id. For example querying for sources for the landslide with id 173:

```
SELECT Source.date, Source.name
FROM  Source JOIN Landslide ON Landslide.id = Source.id
WHERE Landslide.id = 173
```

4.2.2 Queries relating to rainfall data

The rainfall data can be searched by:

- Number of rainfall in mm. This search returns rainfalls that are between two values that the user defines. Search by time interval, which is the same search as for Landslide.
- Cumulative rainfall where the user inserts two of the following three values: cumulative rainfall in mm, the number of antecedent days and the return period.
- Days with landslides that returns all dates that have at least one landslide.
Search for rainfall data where the amount of rainfall is restricted
An example of such a query is the following query retrieving rainfall data where the amount of rainfall is greater than a defined value, in this case 1 mm.

```
SELECT * FROM RAINFALL
WHERE triggering_rainfall > 1
```

Search for rainfall data at some time or time interval
Depending on what search field the user writes in the query will be:
For a specific date; return the rainfall data where the date matches the search word.

```
SELECT * FROM RAINFALL
WHERE date = '1992-06-01'
```

For a year and a month; return the rainfall data where date match the year search word concatenated with the month search word.

```
SELECT * FROM RAINFALL
WHERE date >= '1992-06-01' AND date <= '1992-06-31'
```

Search for rainfall data when a landslide occurred
Retrieves the rainfall data where the number of landslides is greater than zero.

```
SELECT * FROM RAINFALL
WHERE slides > 0
```

4.3 Development of the Graphical User Interface

The developed user interface is divided into three tabs: the landslides search tab, the rainfall search tab and the map search tab. In the map and landslides search tabs the user can search for landslides data, and in the rainfall search tab the user can search for rainfall data. A flow chart illustrating the procedure of searching in the different search tabs is shown in Flow chart 1.
4.3.1 The landslide search tab

The landslides search tab of the GUI is shown in Fig. 10. The search option fields in the landslides search tab are located on the left side of the user interface. Beginning from the top the user can enter the name of a location, or choose in the location list to search for all landslides that occurred in that area. In the interval option he or she can also enter a year, month and/or day to search for the landslides that occurred during that time. It is also possible to search between two dates. The next search option is the level of damage, which is a figure between 1 and 5, which the user defines.

Finally at the bottom there is the keyword search option that lets the user to search by a keyword such as 'Hurricane' or 'School'. The keyword will be matched against the cause, the damage to public infrastructure caused by landslides and the material or movement columns of all landslides. The user can also choose a keyword from a list.
Figure 10. The landslides search tab

On the right side of the landslides search tab there are some options related to the time of the landslides. It lets the user to search, for example, only landslides for a certain time of the day, or a less specific like 'morning', or only landslides with no time recorded.

All of the search options can be combined. For example a user can search for landslides that occurred in the year 2005, with a level of damage equal to 3 and with the time of day both with values like '15:00' and values like 'evening'. When the user has specified what he/she wishes to search for, he/she clicks the OK button. Clicking on the OK button generates an SQL query that is sent to the database for execution. The syntax of the SQL query (for queries that does not use text search, see chapter 3.4) looks as follows:

```
SELECT Landslide.id as id, date, time, location, movement, material, cause, level_of_damage
FROM Landslide  LEFT OUTER JOIN Type ON Landslide.id = Type.id  WHERE [condition 1] [ AND [ condition n ] ]
```

The query is executed and the query result is then displayed in a table in the middle of the search tab (see Fig. 11). If the user wishes to view the newspaper articles that mention a landslide he/she can click on a row in the landslides result table pane to display the name and date of the newspaper article (see Fig. 12).
Figure 11. The landslides search tab after a search

Figure 12. Viewing sources for a landslide
4.3.2 The map search tab

The map search tab lets the user to search for landslides by clicking on a map instead of writing a name of a location. The map is divided into the neighborhoods of Tegucigalpa as shown in Fig. 13. When clicking on a neighborhood the name of the neighborhood and the number of landslides that occurred there is displayed, as well as a list of the names of the locations that are stored in the resulting landslides. To see more information about the landslides in the chosen neighborhood the user has to go to the landslides search tab, where such detailed information is displayed in a table (see Fig. 14). This is done by generating and sending a query to the database for each neighborhood. The syntax of the query for a neighborhood:

```
SELECT Landslide.id as id, date, time, location, movement, material, cause, level_of_damage
FROM ((SELECT id as ts_id FROM Text_search WHERE location MATCH [ location input])
JOIN Landslide ON ts_id = Landslide.id LEFT OUTER JOIN Type ON Landslide.id = Type.id)
```

![Figure 13. The map search tab](image-url)
4.3.2 The Rainfall search tab

The rainfall search tab is shown in Fig. 14. The search options are located to the left in the tab. They are divided into three options; 'Daily Rainfall', 'Interval' and 'Cumulative Rainfall'.

In the daily rainfall option the user can search for dates that had an amount of rainfall either between two values that the user enters, or greater than a value or less than a value. The user can specify a time interval in the same manner as the interval search in the landslides search tab.

The cumulative rainfall option has three input fields: intensity, number of antecedent days and return period. The user specifies either the intensity and number of antecedent days or number of antecedent days and the return period. The missing value will then be displayed in the field to the right of the input field, as will the other values that the user did put in (see Fig. 15).

When the user has specified the search criterion a query will be generated and sent to the database for execution. The syntax of such a query is:

```
SELECT * FROM Rainfall WHERE [condition 1] [ AND [ condition n ] ]
```

The query result, i.e. the resulting dates and the amount of rainfall that fell that day will be displayed in the table in the middle of the search tab. For each date the accumulated rainfall before the date will be displayed at certain intervals (see Fig. 15).
Figure 14. The rainfall search tab

Figure 15. Search result in the rainfall search tab
5. Performance tuning

This chapter describes the changes that were made to the database to optimize the performance of the queries. The columns that the application queries are date, earliest_date_reported, location, time, level_of_damage, cause and damage_to_public_infrastructure from the Landslide table. In the Type table material and movement are queried, as well as date, triggering_rainfall and slides from the Rainfall table.

5.1 Text search

Using text search instead of the ‘like’ operation optimized the queries where a string match is needed, which is on location, cause and damage_to_public_infrastructure. To enable text search in SQLite, the source code for SQLite had to be recompiled with the option that enables virtual tables. Then a virtual table containing the id, cause, location and damage_to_public_infrastructure columns from the Landslide table was created.

A virtual text search table in SQLite contains an index of every word in the columns. The values in the columns are stored as text, ignoring the data types in the create table statement. Using the ‘match’ operation searches the virtual text search table. Match will check the index to see if the word searched for is in the word index.

5.2 Creating indices

The columns that are not used in text search or are included in a primary key are date, earliest_date_reported, time and level_of_damage from Landslide, movement and material in Type and triggering_rainfall and slides from Rainfall. Indices were created for these columns see Appendix I. An index on some of these columns might not increase the execution time for a query on them because they contain a high amount of duplicates. An example is the level_of_damage column in Landslide where the values are either 1,2,3,4 or 5. For those columns a full table scan would be more efficient than fetching rows one by one by index.

SQLite does not gather information about the content of a table by default, but by running the Analyze command, SQLite creates a table that stores statistics about tables and indices, including the amount of duplicate values. SQLite can then use the statistics to determine which index to use. The next chapter contains an evaluation of the effect that the creation of an index has on the performance on columns with and without duplicates. It also examines the effect of running the Analyze command on an index for a column with many duplicate values.
5.3 Prepared queries

Another way of reducing execution time is to use prepared statements. In Java this is implemented with the PreparedStatement class. Prepared statements were used for the retrieval of sources and when calculating the antecedent rainfall for each date in the rainfall search tab. The queries on the landslides and rainfall data might vary from one search to the other, depending on what search options the user chooses, so a prepared statement is not helpful.
6. Measurement Results

This chapter evaluates the impact of the optimization on the database. Measuring the time for executing the queries in the application carried out the tests. The `getCurrentThreadCpuTime()` function in Java was called before and after each call and the time difference was recorded. The `getCurrentThreadCpuTime()` function returns the total CPU time for the current thread in nanoseconds [4]. Each call was made 1000 times to get the average execution time. The changes that were measured were the use of text search instead of the like operation, the creation of indices on date in the Landslide table and on triggering_rainfall in the Rainfall table.

The effect that the statistics table has on performance was measured by executing a query with constraints on `level_of_damage` and `date`. The performance was measured before and after running the Analyze command, which creates the statistics table.

The measurements were made on:

- X64 based PC with processor: AMD Athlon II P320 Dual-Core, 2100 MHz and Windows 7 installed. Memory size 4GB.
- Database size 586 KB in SQLite.
- Buffer cache size 1.95 MB in SQLite.
- The following queries were run on a warm database which is when the cache has data from previous queries in it:
  
  Q1: SELECT * FROM Landslide WHERE location like  
  
  Q2: SELECT * FROM Text_search WHERE location MATCH 'Era ' 
  
  Q3: SELECT * FROM Landslide WHERE level_of_damage = 5 
  
  Q4: SELECT * FROM Rainfall WHERE triggering_rainfall <= 44 
  
  Q5: SELECT * FROM Landslide WHERE date = '2005-01-01' and level_of_damage = 3 

Each query was run 1000 times with prepared queries and the average execution time was taken. The execution time is measured in milliseconds.

<table>
<thead>
<tr>
<th>Optimization criteria</th>
<th>Performance (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Like Q1</td>
<td>0,9</td>
</tr>
<tr>
<td>Match Q2</td>
<td>0,4</td>
</tr>
<tr>
<td>Without index on Q3</td>
<td>0,2</td>
</tr>
<tr>
<td>With index on Q3</td>
<td>0,2</td>
</tr>
<tr>
<td>Without index on Q4</td>
<td>1,7</td>
</tr>
</tbody>
</table>
Table 2. The measurement results

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>With index on Q4</td>
<td>0,2</td>
</tr>
<tr>
<td>Before statistic on Q5</td>
<td>0,4</td>
</tr>
<tr>
<td>After statistic on Q5</td>
<td>0,2</td>
</tr>
</tbody>
</table>

The results in Table 2 shows that the match operation increased the performance with about 50% since the database manager does an index search instead of traversing every word in the column. The index on the level_of_damage had no effect on the performance. The reason for this is as was stated in the previous chapter that level_of_damage contains a high amount of duplicate values, which makes an index less useful. The creation of an index on the triggering_rainfall column increased the performance with 88%.

The impact that the storing of statistics can have on performance is illustrated by the last test. The level_of_damage column and the date column each had an index before this test began. Before the Analyze command the choice of which index to use was arbitrary, since the SQLite optimizer had no information about the indices. If the index on level_of_damage is chosen SQLite has to examine the set of rows where level_of_damage is 3, and then examine the date values. The 'explain query plan' command in SQLite shows what indices SQLite will use to execute a query. When running the explain query plan command before the Analyze command had been run the index that were chosen was the level_of_damage index. The explain query command were run after the Analyze command had been run as well, it then showed the index on the date column. After the Analyze command is run SQLite will pick the index on the column with the lowest amount of duplicates and in this case improve performance with 50%.

Given the size of the database, the changes in performance are hardly noticeable for a user. If the database were to be extended though these changes would make a greater difference, since the use of an index on a column means that the time to find a certain value in that column increases logarithmically instead of linearly. Thus, the use of match instead of like and the creation of indices guarantee that the database application can perform well after an extension of the stored data.
7. Related work

There are a number of landslide databases available online, from all over the Globe. Two examples are the Australian governments landslides database [5] shown in Fig. 16 and the Irish landslides database [6] shown in Fig. 17.

The search interface in the related application is most often implemented as a map that can be navigated by the user, and with the landslides indicated on it as dots or colored areas as shown in Fig. 16. The interface can include other text search fields and result windows to display the info.

The displayed details of the landslides data varies but includes id, date, place, type and cause, and sometimes source and damage. The landslides often have more attributes than these basic ones, for instance the Irish database that stores data about the terrain, what material the terrain consists of, possible vegetation and the angle of the slope.

Figure 16. Interface of the Australian government’s online landslides database
The large difference between the developed Landslide application in this work and the existing online databases is that the graphics are more developed in the other databases, and that the map interface is the main way to search. On the contrary the user interface in this application collects all the search criteria in one place, which makes them easy to survey. The amount of search criteria that the user can specify is also higher than in for example the user interface of the Australian database. The option to search for keywords is very convenient but it doesn't exist in the Australian GUI. The main advantage though is the possibility to correlate the antecedent rainfall with the landslides, and to be able to specify the amount of rainfall and cumulative rainfall. There is no data about the rainfall displayed on either the Australian or the Irish landslides databases.

![Image: Interface of the GSI online landslides database interface (Ireland)](image)

*Figure 17. Interface of the GSI online landslides database interface (Ireland)*
8. Conclusions and future work

The specification for the application was met: the data was organized and stored in a database and a user-friendly application was developed to query it. The execution time of the queries in the database layer of the application is of the magnitude a tenth of a millisecond, which a user would experience as immediate. The application is portable to any system with Java and requires no knowledge of any database concepts to use.

The application could easily be expanded to include more attributes such as for example the angle of the slope. They would either be an expansion of an already existing concept or add a new one. The application interface, though less sophisticated, contains the same functionality as many of the online database search interfaces.
References


   http://docs.oracle.com/database/121/TGSQL/toc.htm

   http://docs.oracle.com/javase/7/docs/api/java/lang/management/ThreadMXBean.html


   http://www.sqlite.org/optoverview.html

Appendix I.  Create statements for the tables

```sql
CREATE TABLE Landslide(
    id INTEGER PRIMARY KEY,
    date TEXT,
    location TEXT,
    time TEXT,
    cause TEXT,
    additional_info TEXT,
    start_of_rainfall TEXT,
    duration_of_rainfall INT,
    deaths INT,
    injured INT,
    homeless INT,
    evacuated INT,
    families_at_risk INT,
    houses_affected INT,
    houses_destroyed INT,
    wall_collapsed INT,
    damage_to_public_infrastructure TEXT,
    level_of_damage INT,
    earliest_date_reported TEXT
);
CREATE INDEX dat on Landslide(date);
CREATE INDEX tim on Landslide(time);
CREATE INDEX lod on Landslide(level_of_damage);
CREATE INDEX edr on Landslide(earliest_date_reported);

CREATE TABLE Source(
    id INT,
    name TEXT,
    date INT,
    FOREIGN KEY(id) REFERENCES Landslide(id)
    PRIMARY KEY (id,name,date)
);
CREATE INDEX date on Source(date);

CREATE TABLE Rainfall(
    date ASC TEXT PRIMARY KEY,
    triggering_rainfall REAL,
    slides INT
);
CREATE INDEX allR on Rainfall(date,triggering_rainfall,slides);
CREATE INDEX trigg on Rainfall(triggering_rainfall);
CREATE INDEX slide on Rainfall(slides);

CREATE TABLE Type(
    id INTEGER,
    movement TEXT,
    material TEXT,
    FOREIGN KEY(id) REFERENCES Landslide(id)
    PRIMARY KEY (id,movement,material)
);
CREATE INDEX mov on Type(movement);
CREATE INDEX mat on Type(material);

CREATE TABLE Reliability_of_Publishing_House(
    name TEXT, id INT,
    when INT, where INT,
    damages INT, causes INT,
    images TEXT, follow_up TEXT,
    exclusive TEXT,
FOREIGN KEY(id) REFERENCES Landslide(id),
FOREIGN KEY(name) REFERENCES Source(name),
PRIMARY KEY (name,id)
);

CREATE VIRTUAL TABLE Text_search USING fts4(
    id INT,
    location TEXT,
    cause TEXT,
    damage_to_public_infrastructure TEXT,
    FOREIGN KEY(id) REFERENCES Landslide(id)
);

Appendix II. Contents of the Excel files

The different columns representing the properties of a landslide stored in the Excel file are described below.

**ID column**
The leftmost column is called id is a unique number assigned to each landslide. This value is used in other parts of the research material of Elias Garcia-Urquia, therefore it should be kept when designing the database.

**DATE and EARLIEST DATE REPORTED columns**
The next two columns are called DATE and EARLIEST DATE REPORTED. DATE is the date when the landslide occurred. If this date is unknown its value is NULL and there will be a date value in the EARLIEST DATE REPORTED column, containing the first date that the landslide was reported in a newspaper article.

**MONTH, MONTH_NUMBER and YEAR columns**
The values for the MONTH and YEAR columns are derived from the DATE value. MONTH_NUMBER is computed as the number of the months between the date and the beginning of the year 1980, i.e. for 1980-January the MONTH_NUMBER is 1.

**SINGLE VRS. MULTIPLE column**
The SINGLE VRS. MULTIPLE column denotes whether there were more than one landslide that DATE. The value is S in if the Date value is unique, M otherwise. Sometimes a C follows the S or M. This means that the landslide continued during the following days. If there is a landslide within seven days in the same locations the original landslide the entry count as continued. This value is composite (single/multiple, continued) and derived.

**SOURCE column**
The SOURCE column contains a list of newspapers that mention the landslide. A name and a date describe a newspaper. The value of the column is both multivalued, as many newspapers can mention the landslide, and composite (name, date).

**LOCATION column**
This column is the name of the neighborhood where the landslide happened. Tegucigalpa is divided into 481 neighborhoods. The naming conventions of neighborhoods in Tegucigalpa mean that multiple neighborhoods can have the same name. They are distinguished by a prefix that can be ‘Col.’ which stands for ‘colonial’, ‘Res.’ which stands for ‘residential’ or ‘Bo’ that stands for ‘barrio’. For example Col. Cerro Grande and Bo Miramesí. In some instances a neighborhood can be divided into two and called upper and lower.

The newspapers are often vague about which exact neighborhood is affected and precludes the prefix. This means that the place names in the landslides table are ambiguous. The current solution is to return all landslides where a part of the location name matches. A search for 'Chile'
will return landslides with location value “Berrinche, close to El Chile Bridge” as well as location values of “El Chile”. The place names are not only names of neighborhoods but also names of roads in a specific neighborhood or just names of roads. Sometimes they can be the name of a hill or a specific place such as a school or other area.

**START OF RAINFALL and DURATION OF RAINFALL columns**
START OF RAINFALL contains the exact time of the day that the rainfall started, or an interval during which it started. DURATION OF RAINFALL denotes in hours or hours and minutes how long the rainfall lasted. These two values are collected from the string value in the ADDITIONAL RELEVANT INFORMATION column.

**RAINFALL INTENSITY and DAILY RAINFALL TONCONTIN columns**
These two columns are added from the rainfall Excel document for convenience and should be removed since they are redundant.

**TIME OF SLIDE column**
This is the time of day that the landslide occurred. Sometimes it is a string like ‘evening’ and sometimes it is an exact time like 14:00 or an interval like 15-16. This attribute is called ‘time’ in the application user interface and in the Landslide table in the database.

**CAUSE column**
This column represents the cause of the landslide and the value of the CAUSE column is a string.

**TYPE OF SLIDE column**
This column denotes the type of the landslide. The type value can be divided into two factors; material and movement. The material can be for example: mud, water, debris, rock and soil, or a combination of these. The movement is divided into five categories according to Varnes’ classification [8]:

- Slide
- Falls
- Spreads
- Topples
- Flows
- Complex (more than one type)

The values in the TYPE OF SLIDE column are composite and multivalued. For example a possible value is “Mudslide and Rockfall”.

**Damage related columns:**
These columns contain the number of deaths, injured people, etc. caused by the landslide, except the DAMAGE TO PUBLIC INFRASTRUCTURE column, which contains a string. The data is collected from the string in the column ADDITIONAL RELEVANT INFORMATION. There are the following columns representing the damages caused by landslides:
DEATHS - the number of deaths caused by the landslide
INJURED - the number of people injured in by the landslide
HOMELESS - the number of people made homeless in the landslide
EVACUATED FAMILIES - the number of families evacuated after the landslide
FAMILIES AT RISK - the number of households endangered by the landslide
HOUSES AFFECTED - the number of houses that were affected
HOUSES DESTROYED - the number of houses that were destroyed
WALL COLLAPSED - the number of collapsed walls
DAMAGE TO PUBLIC INFRASTRUCTURE - contains a string with information about damage to infrastructure.

ADDITIONAL RELEVANT INFORMATION column
This column contains a string that holds the information stored in the damage related columns and the two rainfall related columns START OF RAINFALL, DURATION OF RAINFALL. It might also contain other comments.

NEGATIVE IMPACT (Level of damage) column
The negative impact in the 'NEGATIVE IMPACT' column is a scale that shows the degree of the landslide defined by Elias Garcia-Urquia. It is derived from the damage related columns and the function to calculate it is shown in Fig. 18. It is renamed 'level_of_damage' in the Landslide table in the database and the application.

<table>
<thead>
<tr>
<th>ICN</th>
<th>IMPACT CATEGORY</th>
<th>CRITERIA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Very Low</td>
<td>No damage occurred OR No damage specified.</td>
</tr>
<tr>
<td>2</td>
<td>Low</td>
<td>Damages to the road network.</td>
</tr>
<tr>
<td>3</td>
<td>Intermediate</td>
<td>Less than 10 dwellings affected or destroyed OR Public buildings (health centres, schools, churches) affected.</td>
</tr>
<tr>
<td>4</td>
<td>High</td>
<td>The number of dwellings affected or destroyed is between 10 and 100 OR At least one person injured.</td>
</tr>
<tr>
<td>5</td>
<td>Very High</td>
<td>More than 100 households affected or destroyed OR At least one person killed.</td>
</tr>
</tbody>
</table>

Figure 18. Negative impact
Reliability of the publishing houses columns:
The reliability of publishing houses columns contain weighting factors used to calculate how reliable the material gathered from the newspapers is. The factors are for all the material from a certain publishing house as a whole, not for individual instances of a newspaper. For example, the Excel table has two copies of all the columns (except the column TOTAL), one copy for the publishing house La Tribuna and one copy for El Heraldo. Total is a value to measure the reliability of the data about a landslide collected from the newspapers. There are the following columns representing the reliability of the publishing houses:

WHERE - If location of the landslide was listed the value is 0,3, otherwise it is 0.
WHEN - If the date of the landslide was listed the value is 0,25, otherwise it is 0.
DAMAGES - The number of damages times 0,25.
CAUSES - If the cause was listed the value is 0,1, otherwise it is 0.
TYPE - If the type was listed the value is 0,1, otherwise it is 0.
IMAGES – ‘Y’ if the article contained images, ‘N’ if not, ‘NA’ if not applicable, i.e. if there was no article in the newspaper
FOLLOW UP – ‘Y’ if the newspaper contained one or more articles about the landslide, N otherwise. NA if not applicable, that is if the newspaper had no article about the landslide.
EXCLUSIVE – ‘Y’ if the article focused solely on the landslide, ‘N’ otherwise. ‘NA’ if there was no article.
LT is a derived attribute and is the sum of the weighting factors for the La Tribuna newspaper.
EH is a derived attribute and is the sum of the weighting factors for the El Heraldo newspaper.
TOTAL is a derived attribute and is computed by multiplying the value in the NEGATIVE IMPACT column with (LT + EH).

The Excel landslides table stores only content of the two newspapers La Tribuna and El Heraldo. Even though most of the articles are from those two newspapers there are articles from some other newspapers, for example La Prensa.
### The Landslide Excel table

[Excel table with columns for Location, Source, Month, Year, Number of Casualties, and Other details, with specific data entries for each row.]
<table>
<thead>
<tr>
<th>Course</th>
<th>Type of Lecture</th>
<th>Time of Lecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Theory</td>
<td>10:00-12:00</td>
</tr>
<tr>
<td>456</td>
<td>Seminar</td>
<td>14:00-16:00</td>
</tr>
<tr>
<td>789</td>
<td>Workshop</td>
<td>16:30-18:30</td>
</tr>
<tr>
<td>012</td>
<td>Laboratory</td>
<td>19:00-21:00</td>
</tr>
<tr>
<td>345</td>
<td>Studio</td>
<td>22:00-24:00</td>
</tr>
</tbody>
</table>

**Notes:**
- Theory: Presentation of new concepts and ideas.
- Seminar: Discussion and analysis of selected topics.
- Workshop: Hands-on practice and skill development.
- Laboratory: Experimentation and observation.
- Studio: Performance and critique.

**Requirements:**
- Participation in all lectures is mandatory.
- Attendance and punctuality are crucial for successful completion of the course.
- Active engagement and contribution are expected in all types of lectures.

**Grading System:**
- 4.0: Excellent performance.
- 3.5: Above average performance.
- 3.0: Average performance.
- 2.5: Below average performance.
- 2.0: Poor performance.

**Course Objectives:**
- Understand the fundamental concepts of the subject.
- Develop critical thinking and problem-solving skills.
- Apply theoretical knowledge to practical situations.
- Foster a collaborative learning environment.

**Course Evaluation:**
- Quizzes: 20%
- Midterm Exam: 30%
- Final Exam: 50%
- Participation: 10%
| Case Type | Case ID | Date | Status | Follow-up | Days | Onset Date | Onset Type | Onset Var | Onset Var Type | Onset Var Days | Onset Var ID | Onset Var Status | Onset Var Follow-up | Onset Var Follow-up Days | Onset Var Follow-up ID | Onset Var Follow-up Status |
|-----------|---------|------|--------|-----------|------|------------|------------|-----------|---------------|----------------|-------------|----------------|------------------|----------------------|------------------------|--------------------------|--------------------------|
| Total     |         |      |        |            |      |            |            |           |               |                |             |                |                  |                      |                        |                          |
| India     |         |      |        |            |      |            |            |           |               |                |             |                |                  |                      |                        |                          |
| USA       |         |      |        |            |      |            |            |           |               |                |             |                |                  |                      |                        |                          |
The Rainfall Excel table

<table>
<thead>
<tr>
<th>DAY</th>
<th>DATE</th>
<th>YEAR</th>
<th>MONTH</th>
<th>Day</th>
<th>7 DAY</th>
<th>15 DAY</th>
<th>30 DAY</th>
<th>60 DAY</th>
<th>90 DAY</th>
<th>120 DAY</th>
<th>180 DAY</th>
<th>270 DAY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1980</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1981</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1982</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1983</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1984</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1985</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1986</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1987</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1988</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1989</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1990</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The table contains columns for different days and years, with numerical data in each cell.