Methods for the Identification of Data Outliers in Interactive SQL

Ronald Dattero
Florida Atlantic University

Edna M. White
Florida Atlantic University

Marius A. Janson
University of Missouri-St. Louis

The purpose of this paper is twofold. First, the paper discusses the importance of identifying data outliers (that is, extremely unusual values) for three major purposes in the management of database systems: (1) data validation, (2) statistical analysis, and (3) extreme point recognition. Outliers are defined following the original approach developed by Tukey for use in Exploratory Data Analysis. Second, the paper develops two outlier identification procedures that can be directly inserted into relational database systems through the nonprocedural relational database language SQL (Structured Query Language). The first procedure, the Hinge Procedure, follows exactly Tukey’s definition of outliers, but requires the creation of an additional base table. The second procedure, the Quartile Procedure, provides only a close approximation to Tukey’s definition, but can be implemented solely through views. The development of these procedures was non-trivial due to the rather limited number of available mathematical functions in SQL. The advantages and disadvantages of the two procedures are discussed in the paper.

An outlier can be roughly defined as an extremely unusual data value. Since outliers may be the result of chance, error, or unusual circumstances, it is extremely important to be able to locate and identify any such unusual values when examining, analyzing, or processing a data set. Outliers need to be identified for three distinct, though interrelated, purposes: (1) outliers may aid in data validation by identifying data points that are so extreme that there is a high probability of input error, (2) outliers may be removed from statistical analysis because they may have an adversely high impact on various measures such as the mean, and (3) outliers may be associated with extremely good or bad performances that should be recognized. All three purposes of outlier identification are important in the management of database systems, particularly when the database is used to support decision making.

Many methods exist for the identification of outliers. However, these procedures are external to the database system (often in statistics packages) since commercial database systems do not provide the necessary procedures. Outlier identification is typically not performed due to the effort required (albeit small) to export data to the statistics package combined with a limited appreciation of the importance of data outliers. Simple integrity constraints, such as data type and range checks, may be employed. However, although such constraints will reduce (though probably not eliminate) the acceptance of invalid data values ((1) above), these constraints do not address the other two issues.

The purpose of this paper is two-fold: to provide an overview of the importance of data outliers and to develop procedures that can be directly inserted into relational database systems. This paper restricts its focus to relational database systems because almost every modern database product is based on the relational data model (Date, 1990). The procedures will be developed in the relational database language SQL (Structured Query Language) because it is the established standard language for relational databases.

SQL is a nonprocedural (also called descriptive or declarative) language. The only mathematical functions that are standard in SQL are: COUNT, MIN, MAX, SUM, and, AVG (which are, respectively, the count, minimum, maximum, sum, and average functions). These limitations of SQL have lead some authors, such as Lans (1988), to conjecture that even “determining the median of a column with duplicate values is not possible with interpretative
SQL”. In fact, although not impossible (as demonstrated by this paper) with nonprocedural languages, the development (though not the use) of such methods is substantially more difficult than through standard procedural languages.

In the next section the definition of outliers will be formally developed and the importance of outlier identification will be addressed. Next, the development of the two procedures for outlier identification through SQL will be detailed, and an application will be presented. The first procedure presented, the Hinge Procedure, follows Tukey’s (1977) method of outlier identification exactly, but requires the creation and storage of an additional base table in the database to support roundup and rounddown capabilities. The creation of an additional base table (that is, a table that is physically stored) may make this method less attractive to some users. The second procedure, the Quartile Procedure, employs only views making the development of the procedure somewhat more difficult. As a consequence, the Quartile Procedure employs an approximation of Tukey’s method. A formal argument supporting the approximation is supplied in the appendix. The final section provides a comparison of the two procedures and the conclusions of this paper.

Outliers

Hartwig and Dearing (1979) provide the following definition of an outlier: “An outlier is a value which lies outside the normal range of the data, i.e., lies well above or well below most, or even all, of the other values in a distribution.” Any set of data may have zero, one, or multiple outliers; each of the outliers may be a valid, though unusual, value, or the result of an error. Every set of data should be examined for the presence of outliers. “Whatever their source, outliers demand and deserve special attention. Sometimes we will try to identify and display them; other times we will try to insulate our analyses and plots from their effects” (Velleman and Hoaglin, 1981). As previously stated, outlier identification can aid in data validation, statistical analysis, and extreme point recognition.

Data Validation

Any data set may contain incorrect values caused by measuring, recording, copying or inputting error. When any such errors occur, they should be detected and corrected. However, errors that result in outliers (for example, 19 transposed to 91) are likely to have more unfortunate consequences than errors that do not result in outliers (for example, 19 read as 17). Note that a data type check, such as DECIMAL (2) in SQL/DS (Date and White, 1989), would not detect the error of transposing 19 into 91. Before using any data, the validity of any outlying values should be carefully examined (Velleman and Hoaglin, 1981).

As an example in the use of outlier identification for data validation, consider the data set shown in Figure 1. The value 100 is clearly an outlier from the rest of the data. Once the outlier has been identified, common sense and experience suggest that the decimal point was omitted from the value 10.0 — a suggestion that can be easily checked. Data validation is simply the process of detecting and correcting data errors. Outlier identification is a quick and effective method for selecting values that are important for closer examination and validation. Recently, Janson (1988) employed exploratory data analysis techniques, including outlier identification, to address the data validation problem.

Statistical Analysis

Suppose the data value 100 shown in Figure 1, whether correct or incorrect, is allowed to remain in the data during further analysis. The impact of such an extreme data point has an enormous impact on summary measures such as the mean. In this example, the mean is 19.99 with the value 100 included and only 6.65 if 100 is excluded (or 7.13 if the 100 is changed to 10.0). This is a good example where “one

| Data Set: 4.7, 5.1, 5.9, 6.4, 8.7, 9.1, 100 |  |
|---|---|---|
| **Median:** | **Tukey’s Hinges:** | **Quartile Procedure:** |
| Lower Hinge/Quartile: | 5.5 = (5.9 + 5.1)/2 | 5.1 |
| Upper Hinge/Quartile: | 8.9 = (8.7 + 9.1)/2 | 9.1 |
| H-Spread/Q-Spread: | 3.4 = 8.9 - 5.5 | 4.0 = 9.1 - 5.1 |
| Step: | 5.1 = 1.5 * 3.4 | 6.0 = 1.5 * 4.0 |
| 2 Steps: | 10.2 = 3 * 3.4 | 12.0 = 3 * 4.0 |
| Lower Inner Fence: | 0.4 = 5.5 - 5.1 | -0.9 = 5.1 - 6.0 |
| Upper Inner Fence: | 14.0 = 8.9 + 5.1 | 15.1 = 9.1 + 6.0 |
| Lower Outer Fence: | -4.7 = 5.5 - 10.2 | -6.9 = 5.1 - 12.0 |
| Upper Outer Fence: | 19.1 = 8.9 + 10.2 | 21.1 = 9.1 + 12.0 |
| Points Outside: | None | None |
| Points Far Out: | 100.0 | 100.0 |
Related Content

INDUSTRY AND PRACTICE: Research Issues in Multimedia Databases
www.igi-global.com/article/industry-practice-research-issues-multimedia/51171?camid=4v1a

An Empirical Investigation of the Perceived Benefits of Agile Methodologies Using an Innovation-Theoretical model
www.igi-global.com/article/an-empirical-investigation-of-the-perceived-benefits-of-agile-methodologies-using-an-innovation-theoretical-model/172453?camid=4v1a

Learning Classifiers from Distributed Data Sources
www.igi-global.com/chapter/learning-classifiers-distributed-data-sources/20744?camid=4v1a

Ensuring Correctness, Completeness, and Freshness for Outsourced Tree-Indexed Data
www.igi-global.com/chapter/ensuring-correctness-completeness-freshness-outsourced/8031?camid=4v1a