Scaling Out the Discovery of Inclusion Dependencies

Sebastian Kruse, Thorsten Papenbrock, Felix Naumann

Hasso Plattner Institute
Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam
firstname.lastname@hpi.de

Abstract: Inclusion dependencies are among the most important database dependencies. In addition to their most prominent application – foreign key discovery – inclusion dependencies are an important input to data integration, query optimization, and schema redesign. With their discovery being a recurring data profiling task, previous research has proposed different algorithms to discover all inclusion dependencies within a given dataset. However, none of the proposed algorithms is designed to scale out, i.e., none can be distributed across multiple nodes in a computer cluster to increase the performance. So on large datasets with many inclusion dependencies, these algorithms can take days to complete, even on high-performance computers.

We introduce SINDY, an algorithm that efficiently discovers all unary inclusion dependencies of a given relational dataset in a distributed fashion and that is not tied to main memory requirements. We give a practical implementation of SINDY that builds upon the map-reduce-style framework Stratosphere and conduct several experiments showing that SINDY can process huge datasets by several factors faster than its competitors while scaling with the number of cluster nodes.

1 Discovering Inclusion Dependencies

Given a relational dataset, an inclusion dependency (or short IND) is a constraint expressing that all values of some column (combination) A are also found in some other column (combination) B. This is formally expressed as $A \subseteq B$ and read as "A is included in B". In the context of this IND, A is called the *dependent* column and B the *referenced* column.

This rather simple type of dependency has a broad range of applications, the most prominent one being foreign key discovery. In a dataset that consists of multiple relations, foreign keys describe how their records relate to each other, thus, their knowledge is crucial to any further work with this dataset. However, for various reasons this knowledge can be lacking: datasets are obtained as dumps in some export format that does not capture the schema of the dataset but merely its content; not all databases are capable of enforcing foreign key constraints; and even if so, capturing these constraints is sometimes avoided for performance reasons as the foreign key enforcement involves some computational overhead. Besides foreign keys, INDs are valuable knowledge to many other applications [CTF88, Gry98, LV00], like schema redesign (e.g., detect duplicate information) and data integration (e.g., joining tables from multiple data sources).

As opposed to the detection of other data dependencies, such as functional dependen-