# Fast and Dynamic OLAP Exploration Using UDFs

Zhibo Chen
University of Houston
Dept. of Computer Science
Houston, TX 77204, USA

Carlos Ordonez
University of Houston
Dept. of Computer Science
Houston, TX 77204, USA

Carlos Garcia-Alvarado
University of Houston
Dept. of Computer Science
Houston, TX 77204, USA

## ABSTRACT

OLAP is a set of database exploratory techniques to efficiently retrieve multiple sets of aggregations from a large dataset. Generally, these techniques have either involved the use of an external OLAP server or required the dataset to be exported to a specialized OLAP tool for more efficient processing. In this work, we show that OLAP techniques can be performed within a modern DBMS without external servers or the exporting of datasets, using standard SQL queries and UDFs. The main challenge of such approach is that SQL and UDFs are not as flexible as the C language to explore the OLAP lattice and therefore it is more difficult to develop optimizations. We compare three different ways of performing OLAP exploration: plain SQL queries, a UDF implementing a lattice structure, and a UDF programming the star cube structure. We demonstrate how such methods can be used to efficiently explore typical OLAP datasets.

## Categories and Subject Descriptors

E.1 [**Data Structure**]: [Trees]; H.2.4 [**Database Management**]: Systems—*Relational Databases*; H.2.8 [**Database Management**]: Database Applications—*Data Mining*

## General Terms

Algorithms, Experiments, Measurement

## Keywords

OLAP, UDF, CUBE

## 1. INTRODUCTION

On-Line Analytical Processing (OLAP) [1] is a set of techniques that allow the user to efficiently retrieve specific aggregations from large datasets with multiple dimensions. OLAP exploration and analysis is often performed through the use of precomputed subcubes or by transforming the dataset into a different structure from which subcubes can be quickly obtained. Regardless of the method, results are still obtained either from an OLAP server or from a source outside the database management system. The problem with these approaches is that additional OLAP servers are costly and require additional maintenance, while exporting large datasets from the DBMS takes a long time.


http://doi.acm.org/10.1145/1559845.1559989

The main motivation behind our research lies in observing the viability of performing OLAP techniques entirely within a database management system, without the need to build a separate OLAP server or to export the dataset. Thus, we not only reduce the cost of installing and maintaining additional OLAP servers, but also remove the delay of exporting datasets. In addition, we can enjoy the benefits offered by database systems, such as improved security, fault tolerance and querying with SQL. We found that by using user-defined functions (UDFs), we can solve OLAP operations faster than using standard SQL and can efficiently extract subcubes from typical OLAP datasets. In addition, for large datasets, the time to export a dataset and import results makes performing OLAP operations outside the database inefficient. In this demonstration, we will show how we can explore OLAP cubes as well as perform some simple analysis on the subcubes. In addition, we will show how additional knowledge can be extracted through the combination of OLAP and parametric statistical tests [3]. Finally, we will also compare the UDF approach to OLAP techniques with the standard SQL approach, which is known to be slow.

In this article, we provide a technical summary in Section 2 and outline our system demonstration in Section 3.

## 2. TECHNICAL OVERVIEW

We will now give a more detailed analysis of how OLAP techniques can be performed with User-Defined Functions (UDFs). It is important to note that our primary goal is to produce an application with the ability to perform OLAP techniques entirely within a database, without the need to alter any source code. Before we look at the technical aspects, we will discuss some definitions used through this demonstration. We consider a typical OLAP dataset as having $n$ records, $d$ cube dimensions, and $e$ measure attributes. The dimensional lattice is the data structure that represents all the subsets of dimensions and their containment. Nodes represents a particular combination of dimensions in the lattice. We can further separate nodes into groups, which represents a specific combination of dimension values.

### 2.1 OLAP Operations

OLAP queries normally fall into two categories: retrieval and aggregational. The retrieval queries involve operations, such as slicing, dicing, and pivoting, that extracts data from specific nodes of the dimensional lattice. These queries primarily use select statements to retrieve the appropriate results. On the other hand, the aggregational OLAP queries,

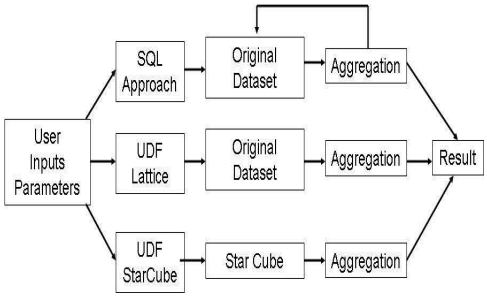


**Figure 1: Design flow of SQL and UDF approaches.**

such as drill up and roll up, requires the creation of entire nodes on the lattice. These queries require group-by statements that often takes longer time than the retrieval queries. For both of these queries, there are three main ways that they can be resolved. The first method involves computing the entire dimensional lattice, storing it in a table, and then retrieving the required tuples as needed. While this method offers some of the fastest response times, it becomes impractical at higher dimensions and also uses the most storage space. The second method creates intermediate subcubes that are strategically places around the entire dimensional lattice. In this way, when a query is inputted, the closest intermediate subcube is determined and used to produce the results. The assumption is that these intermediate subcubes will have less tuples than the original dataset. The third and final method involves calculating all the queries directly from the original dataset. This method uses the least amount of storage space, but is the slowest of the three. The method to use depends on the size of the original dataset. If the dataset has a large $n$ and $d$, then the first method is not viable. On the other hand, if the dataset is small, then it would be practical to simply calculate the entire dimensional lattice. In our demonstration, we consider these three methods to be optimizations and can be controlled by the user.

## 2.2 Standard SQL Approach

The SQL approach involves the use of standard SQL to produce the required results of OLAP queries. For the most part, the SQL required includes at least one aggregation on some particular dimension(s). For the scalar operations, the SQL requires either a straight select to retrieve rows from already computed nodes or an aggregation to compute the required node, followed by a select statement. Both the aggregation and select statements are fairly straight forward, with the dimensions in question being involved in the query. If multiple nodes of the dimensional lattice are required, then an aggregation would need to be performed for each node. For certain databases, there also exists built-in functions, such as grouping sets, which would allow for the aggregation of multiple sets of dimensions. More detailed explanations of the actual queries can be found in our previous research [3].

## 2.3 UDF Approach

Our goal in the UDF approach is to significantly reduce disk access by performing most exploratory operations in main memory through the use of UDFs. The main component behind the UDF is the data structure that needs to be created to store essential information from the dataset. This structure is held in main memory and is used to retrieve the results of the OLAP queries. Currently, we have two different data structures: a lattice-based structure and a starcube-based structure. The lattice-based data structure, shown in detail in our previous research [2], uses a two-tiered design to represent the dimensional lattice. The first level stores the nodes and the set of dimensions represented, while the second level stores data pertaining to specific values of those dimensions. The required OLAP data, such as summation or averages of the cells, are stored directly into the second level. This data structure only requires one pass through the dataset because it has the ability to perform multiple aggregations simultaneously.

A second alternative data structure that we are currently researching is the starcube [4] inside the UDF. The starcube is a data structure that can be used to represent the actual dataset. Once a dataset is changed into this new format, OLAP operations can be quickly retrieved by traversing the structure. Unlike the previous data structure, the aggregations are not stored within the starcube. Instead, an additional pass through the starcube structure is required to obtain the appropriate results. However, the starcube only needs to be computed once, similar to a precomputation of the dataset that is used in the SQL approach. Once generated, the UDF needs to load the starcube and all OLAP operations can then be processed in memory.

For both of the UDF approaches, the optimizations are all related to the C language. This is because the UDF is programmed in C before being built and deployed into the DBMS. One of the optimizations that we used is a while statement instead of recursion when traversing the tree structures. The reasoning is that using recursion would waste too much memory. Several other implementation optimizations were also included, such as the use of strings to represent structures instead of creating a struct.

Regardless of the data structure used within the UDF, the final result of the UDF needs to be a table. In past years, the only common UDF types were scalar and aggregation. For both, the return value is always single values that can only be used to store small amounts of information. Instead, we decided to use Table-Valued Functions (TVFs), a novel type of UDF supported by some DBMS, that allow for the return of an entire table, instead of one tuple per set. This is useful because it avoids the extra intermediate step of having to transform the original dataset into a multi-valued attribute (i.e. long string). With the TVF, we can read in entire datasets and return entire tables. The novelty of using UDFs to solve OLAP operations is the use of previously C-only algorithms within the UDF of a DBMS. We show how starcube can be translated from C into a UDF. To the best of our knowledge, few researchers have studied the viability of performing OLAP algorithms within a DBMS through the use of UDFs instead of an external language, such as C.

# 3. SYSTEM DEMONSTRATION

The general data flow of our application, including the various approaches, is shown in Figure 1. Note how both UDF approaches only use one or two passes of the dataset while the SQL approach requires one pass for each aggregation. For our application, we used Java 1.6 to create the GUI. Since the user is allowed to open up multiple windows in which to perform OLAP operations, we created one

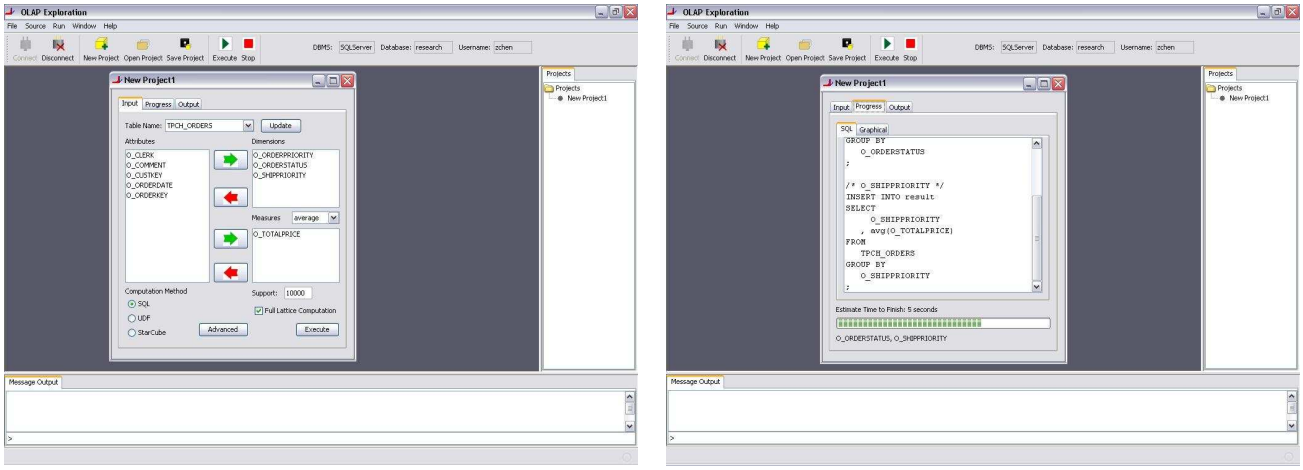


**Figure 2: User Interface of OLAP Exploration and Analysis using DBMS.**

thread for each window. Each thread would have its own connection to the database using JDBC. As a result, the user will be able to execute multiple windows at the same time. When an OLAP operation is executed, the application performs one of the two approaches, depending on user preference. For both approaches, the application will generate the appropriate SQL queries and automatically send the queries to the database. When the queries are completed, the application would automatically retrieve the appropriate results and display them to the user.

For the demonstration, we will use a system running a commercial database system that supports TVFs. We will demonstrate our application, Figure 2, for OLAP exploration and analysis within this DBMS on these two datasets: TPCH-ORDERS and a synthetic dataset. The first dataset comes from the TPC-H benchmark and contains 1M tuples with a small number of dimensions. The second dataset is randomly generated with a large number of tuples (5M) and higher number of dimensions. The number of distinct values for each column will vary from two to five. We will run our demonstration on a local DBMS installed on our laptop and will not require any internet connections.

Our demonstration can be separated into two distinct parts. The first part will demonstrate the viability of performing OLAP operations within a DBMS, while the second portion will compare the various ways in which we can arrive at the correct result. For the first portion of the demonstration, we will use the larger dataset. We will show how the user can easily select the various parameters, optimizations, and the approach with which to generate the results. For example, we will show how a user can load the synthetic dataset and choose several dimensions from the fifty in the list. Then, the user can easily select the various other parameters, such as support, aggregation operation, as well as the approach and the optimizations, such as whether to calculate the full lattice or not. Once the results are obtained, we will show how a user can then navigate throughout the subcube, using standard OLAP techniques such as roll-up, drill-up, and slice. In addition, Figure 2 shows how the user also has the ability to view the SQL code for the standard SQL approach. We can also graphically see the data structure that is stored into main memory by the UDF approach as well as the actual code that generated the structure. Also, we will show how the user can alter the parameters and observe the new results. We will also demonstrate the undo feature, which allows the user to roll back to a previous set of parameters. Afterward, we will demonstrate how the user can open another window to perform separate exploration and analysis on either the same dataset or a different one. The multiple windows allow the user to compare results or to simultaneously perform multiple OLAP operations.

The second portion of the demonstration will compare the performance of the different approaches. We will show the difference in execution time of the different approaches as well as different parameters. Because we will be generating full lattices in our demonstration, we will be primarily using the smaller TPCH-ORDERS dataset. We will demonstrate the difference in execution time between the different approaches when generating entire lattices and partial subcubes. For example, we will compare the execution times of generating the full lattice of the TPCH-ORDERS dataset. Since the dimensionality is low, this portion should not take more than a few minutes. We will also compare the execution times of obtaining a subcube of the lattice. Even though the original dataset might have a high number of dimensions, by choosing a few dimensions, we can still obtain efficient times. The ultimate goal of our demonstration is to convince the user that OLAP operations can feasibly be performed within a DBMS.

## 4. REFERENCES


[1] S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. *ACM SIGMOD Record*, 26(1):65–74, 1997.

[2] Z. Chen and C. Ordonez. Efficient OLAP with UDFs. In *ACM DOLAP*, pages 41–48, 2008.

[3] C. Ordonez and Z. Chen. Evaluating statistical tests on OLAP cubes to compare degree of disease. *IEEE TITB*, 2009.

[4] D. Xin, J. Han, X. Li, and B. Wah. Star-cubing: computing iceberg cubes by top-down and bottom-up integration. In *VLDB*, 2003.