A spatial column-store to triangulate The Netherlands on the fly.

Romulo Goncalves[1], Tom van Tilburg[2], Kostis Kyzirakos[3], Foteini Alvanaki[3], Panagiotis Koutsourakis[4], Ben van Werkhoven[1], and Willem van Hage[1]

[1]NLeSC, The Netherlands
{r.goncalves,b.vanwerkhoven,w.vanhage}@esciencecenter.nl
[2]Geodan, The Netherlands
tom.van.tilburg@geodan.nl
[3]CWI, The Netherlands
{kostis.kyzirakos,f.alvanaki}@cwi.nl
[4]MonetDB Solutions, The Netherlands
panagiotis.koutsourakis@monetdbsolutions.com

ABSTRACT

3D digital city models, important for urban planning, are currently constructed from massive point clouds obtained through airborne LiDAR (Light Detection and Ranging). They are semantically enriched with information obtained from auxiliary GIS data like Cadastral data which contains information about the boundaries of properties, road networks, rivers, lakes etc.

Technical advances in the LiDAR data acquisition systems made possible the rapid acquisition of high resolution topographical information for an entire country. Such data sets are now reaching the trillion points barrier. To cope with this data deluge and provide up-to-date 3D digital city models on demand current geospatial management strategies should be re-thought.

This work presents a column-oriented Spatial Database Management System which provides in-situ data access, effective data skipping, efficient spatial operations, and interactive data visualization. Its efficiency and scalability is demonstrated using a dense LiDAR scan of The Netherlands consisting of 640 billion points and the latest Cadastral information, and compared with PostGIS.

1. INTRODUCTION

Continuous monitoring of manufactured structures in search of small deviations or small breaches, assessment of an urban area re-organization, and under- and over- ground formation analysis are key activities for urban planning, risk management and natural resource management. Such studies are conducted on 3D digital city models, which consist of large collections of semantically rich objects with many properties such as material and color.

3D digital city models are currently constructed from massive point clouds obtained through airborne LiDAR (Light Detection and Ranging) or terrestrial scanning campaigns. They are constructed through segmentation and triangulation of a point cloud thereby creating a surface representation. Their semantical information is obtained from auxiliary GIS data like Cadastral data, which contains information about the boundaries of properties, topographical data that describe buildings, road networks, rivers, lakes, etc.

Technical advances in the LiDAR data acquisition systems made possible the acquisition of high resolution topological information for an entire country. As an example, the topology of the Netherlands, the *Actueel Hoogtebestand Nederland 2* (AHN2) [1] which is stored and distributed in more than 60,000 LAZ files, contains 640 billion points and it has in average 10 points per square meter.

With Cadastral data being constantly updated and LiDAR data being extended with periodic scans, 3D digital city models must be reconstructed periodically, i.e., not being generated anymore with a onetime large pre-computation job, and allow user interaction. Such demands made us to re-think on how spatial computations, data management and data processing is performed in large scale. Our goal is to modernize the generation and manipulation of 3D digital city models by extending a Geospatial Database Management System (DBMS) with all necessary functionality to access directly raw data sets without requiring data to be pre-loaded and take advantage of GPUs to accelerate the 3D digital city models generation thereby vastly improving flexibility and performance.

To step away from traditional Spatial Database Management Systems (SDBMS) which are all record-oriented architectures, known to perform worse than column-oriented architectures, we have extended a modern column-store, MonetDB [11]. Through vertical partitioning of relational tables, column-stores significantly reduce cost of data access. In our case, vertical partitioning is exploited to reduce the number of columns to be imported in the database. Such data organization improves data compression, simplifies data skipping strategies and is well suited for vector processing.

For in-situ data access, the MonetDB geospatial module was extended to access directly large spatial data repositories, i.e., it keeps data in its original format while scalable processing functionality is offered through the DBMS. For efficient spatial selections and near real time 3D model generation some of these operators are comple-

SIGSPATIAL'16 October 31 - November 03, 2016, Burlingame, CA, USA

ACM ISBN 978-1-4503-4589-7/16/10.

DOI: http://dx.doi.org/10.1145/2996913.2997005

mented with a GPU version.

The efficiency and scalability of our work is demonstrated through a web-application, which does on-demand classification of a dense nationwide LiDAR scan of The Netherlands (640 billion points) with Cadastral information. For demonstrating the performance of our approach we use PostGIS boosted by its new Block Range Index (BRIN) [2] as an alternative database solution.

The remainder of the paper is as follows. Section 2 discusses the general architecture. In Section 3 we use multiple use case scenarios, to demonstrate the flexibility and efficiency of generating and exploring 3D city digital models using our approach. Finally, in Section 4 we conclude the paper.

2. ARCHITECTURE

In this section we outline the architectural extensions to a column-oriented SDBMS to provide effective data skipping and efficient spatial processing. The effective data skipping is part of our strategy for *in-situ* data access where data is kept in its original format while scalable and distributed processing functionality is offered through a DBMS. For efficiency purposes we push spatial processing down into the columnar-oriented kernel for exploiting inter- and intra- operator parallelism while certain operators are GPU-accelerated.

2.1 Spatial DBMSs

To efficiently combine heterogeneous data sources each data set should be stored under the same storage without changing the users' data structure perception, i.e., the conceptual schema [3] should remain the same. The clear separation between the physical and the conceptual schema creates the opportunity to have different types of applications exploring the same data sets. Furthermore, data organization at the physical storage system differs from the conceptual schema to optimally exploit the hardware characteristics. Such separation between the physical layout of the data and its conceptual model is what made Database Management Systems (DBMS) so successful for analytic workloads.

Many DBMSs have been successfully extended with support for spatial and geo-spatial applications. For instance, the OGC implementation specification "Simple feature access: SQL option" [7] that defines basic geometry types and operators is followed by PostGIS, Oracle, MySQL, Microsoft SQL Server and MonetDB. Due to the complexity of their software stack, such functionality is provided through user-defined functions (UDF) and augmented in some cases with spatial search accelerators. Deep integration with the database engine is not always possible, therefore, most functionality is brought in by dynamic linkage with external libraries such as GEOS [4] and SFCGAL [5] libraries.

For our work we have extended the column-oriented SDBMS MonetDB [11], which steps away from traditional SDBMSs that are all record-oriented architectures. Through the works [10, 9, 15, 14, 18], MonetDB spatial features have been matured to provide core technology components for geo-spatial analysis of large data sets. Atomic spatial types and their operations are becoming part of the relational kernel and not an add-on. All the operations are available for spatial applications through integrated environments, such as R and Python, and an SQL front-end. The spatial query model that is used by MonetDB follows the well-established two-step approach of **filtering** and **refinement** [9].

By being a column-oriented SDBMS, our solution offers vertical partitioning of relational tables which significantly reduces data access, improves data compression, simplifies data skipping strategies, suits well vector processing and the integration of GPU-accelerated operators (c.f., Section 2.3). In our case, vertical partitioning is further exploited to reduce the number of columns to be imported as we will explain in Section 2.2.

2.2 In-situ data access

Technical advances in the LiDAR data acquisition systems made possible the acquisition of high resolution topology maps of an entire country. As we mentioned earlier, a nationwide LiDAR dataset like AHN2 comprises 640 billion points averaging 6-10 points per square meter. For this dataset, the authors of [17] spent nearly 18 hours for extracting, transforming, and loading (ETL) the dataset into MonetDB because the data was initially converted to the internal format of MonetDB and then imported through a bulk loading operation. The same amount of time was spent for sorting and indexing AHN2 to be able to query it efficiently using the well-known LiDAR file-based solution Rapidlasso LAStools [8].

To reduce such a costly pre-query preparation step we decided to keep data in its original format and access it through a lazy and iterative import process that accesses the raw files directly. This approach is a LiDAR extension of the strategy described in [10] that discusses how to provide *in-situ* data access to large NetCDF data repositories. The general concept was developed in the context of the data-vaults framework [12].

The data-vaults framework defines that data access comprises three phases: attaching a file, importing the file's content, and collecting statistics that boost query optimization. During the attachment phase, the file's metadata is loaded into a special DBMS catalog. At query time, only a sub-set of the attached files is imported according to a fast approximation of the spatial predicates that appear in the query.

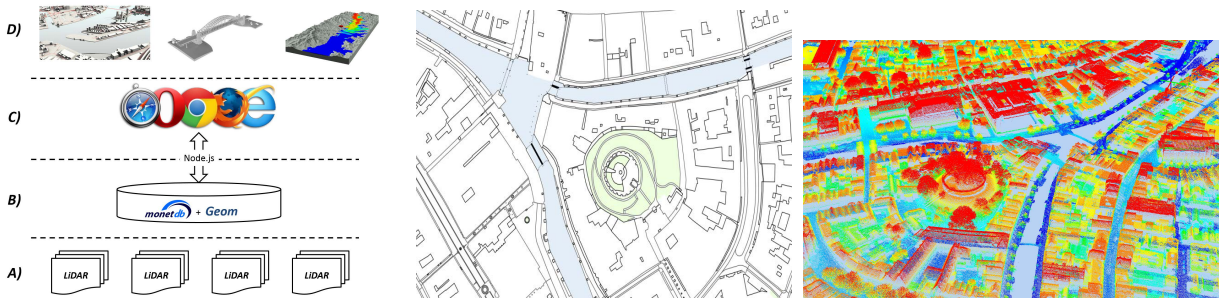
We follow two strategies for importing data. If the file format defines that each attribute is stored sequentially then the import process memory maps each attribute as a column, otherwise, the data is converted and loaded into the database as temporary data. In the latter case, cache policies, such as Least Recently Used (LRU), are used for data eviction. Once the attached files are imported during the filtering step, most of the imported points are identified and disregarded using secondary indexes such as column imprints [16]. Column-imprints resemble bitmaps that index ranges of values in each cache line of each column. This makes them very efficient for evaluating range queries since they allow skipping cache lines that do not contain data for a desired range.

The refinement step operates on the results of the filtering step that produced a superset of the solution. During this step, the spatial predicate is evaluated against the precise geometry G. The refinement step can be very expensive, especially when the geometries are complex. To circumvent the issue, we exploit GPU technology to speed up the refinement step.

2.3 Spatial DBMSs and GPUs

Arithmetic intensity is defined as the number of operations performed per word of memory transferred. Operators with high arithmetic intensity that consume a large stream of points with minimal dependency between data elements should be executed on GPUs. For spatial operations which process many vertices or geometries in the same way, such as point in a polygon (PIP), GPUs provide very effective processing since they can process many points in parallel. On the other hand, spatial operators that rely on algorithms with many branches are better executed on CPUs.

At the best of our knowledge, none of the SDBMS, especially column-oriented SDBMS, have integrated GPU-accelerated operators. All solutions with GPU-accelerated operators are simple standalone file-based libraries which are hard to integrate into existent data workflows or record-oriented architectures.



(a) Architecture (b) Cadastral data (c) LiDAR data

Figure 1: Architecture and data sets

Our work integrates GPU aware spatial primitives into the column-oriented kernel of MonetDB for efficient geo-spatial querying. Within a single query the user has column-store efficient I/O and massive GPU parallelism for spatial operators. MonetDB allows a seamless integration due to two special features of its architecture, the operator-at-the-time paradigm and late materialization. According to the operator-at-a-time execution paradigm, complete intermediates are a by-product of every step in the query execution plan [13]. Hence, the scheduling of an operator for execution only happens when all its arguments have been made available. The feature simplifies the integration of a GPU operator since the intermediate materialization simplifies the data transfer between the main-memory and the GPU memory while the dependencies on the result to be fetched from the GPU are taken care by the scheduling strategy followed by MonetDB.

For late materialization, tuples are re-constructed as late as possible in the query plan. This allows column-oriented architecture to have a low memory footprint during query execution, especially during the filtering phase. Such feature is kept intact by introducing GPU operators that consume and return candidate lists which are then used by the following operators in the query plan.

2.4 Spatial Queries

Queries that involve spatial data are generally more complex than normal relational queries. In this case, the queries describe complex processing steps that transform data into a 3D model. The most expensive operations involve a nearest neighbor search in the point cloud for discovering the elevation of the vertex of each polygon and triangulating the resulting 3D polygons. In addition, a filtering algorithm is applied to the pointcloud data for classifying vegetation points and computing the average roof height. The most important feature of the queries however is finding the intersections between polygons and points for which they make use of indexes

3. DEMONSTRATION

We propose a demonstration with a live interactive experience using a X3D viewer to visualize 3D digital city models of The Netherlands. Using the interface illustrated in Figure 2, the user can select a city, angle of view, resolution, etc.

The data sets for the demo session are the latest topography of The Netherlands, *Actueel Hoogtebestand Nederland 3* (AHN3) [1] and the latest Cadastral information for The Netherlands, Basisregistratie Grootschalige Topografie (BGT) [6]. Our solution outlined in Section 2 is the backend used for their integration, management, and processing. Through three scenarios, the audience will see the benefits of in-situ data access, GPU aware operators and compare its efficiency with PostGIS.

3.1 User interaction

Our solution caters for multiple types of applications; SQL can be used for analytical workloads, but also provides an integrated environment for R and Python. After processing the data our solution has the option to export the results to standard formats to be loaded into visualization tools. Examples are GeoJSON, LAS and X3D, the latter being used by the web-application of our demo. Because the X3D format is able to directly display the triangulated surfaces in a web-browser it serves as a convenient way of sharing output. In our demo, a user expresses an area of interest with a predicate on the semantic attributes she wants to visualize. The request is translated into an SQL query and through a MonetDB node.js module it is pushed down to the remote MonetDB server, layer *D), C) and B)* in Figure 1a.

After query compilation, the database performs all the steps described in Section 2.2 to make the necessary data available, i.e., *in-situ* access to the file repositories in layer *A)* Figure 1a. With all relevant data available, the filtering and the refinement steps are executed. The selected points are then grouped per object and each object is triangulated using Constrained Delaunay Triangulation. The query result is exported as X3D format and sent back to the browser for rendering.

In the web browser, the resulting scene shows a 3D extrusion from the Cadastral data where the terrain is modelled along its vertices and buildings are extruded to their average height. The scene can be rotated and lighting, and colors can be changed. The most important feature however is that the individual entities from the Cadastral data (like roads, houses, bridges) can be highlighted or otherwise individually dealt with. This gives way to more advanced interaction with the database in the future, where users can add information to the data and send it back to the database to run more advanced modelling (e.g. flood or noise modelling).

The users will notice that data processing requires only a few seconds which gives users the option to react instantly to changes in the data. For instance, when results differ from the expected ones, the user can change some query parameters and see the difference almost instantly.

3.2 Data sets

For data integration, we will be using BGT semantic information and AHN3 point cloud data. The BGT data is a detailed 2D digital map of The Netherlands. It contains the location and footprint of objects such as buildings, roads, water, railways, and green areas as illustrated in Figure 1b. The BGT data is freely distributed as open data [1].

AHN3 dataset [1] is the latest LiDAR scan of The Netherlands and it is also freely distributed as open data [2]. The sample density is 6 to 10 points per square meter for the entire country. Figure 1c presents a 3D visualization of the AHN2 dataset. During our demo

[1]https://www.pdok.nl/nl/producten/pdok-downloads/download-basisregistratie-grootschalige-topografie

[2]https://www.pdok.nl/nl/ahn3-downloads

session the latest scan AHN3 is used.

3.3 First scenario

In the first scenario, the audience will be familiarized with the data sets described in Section 3.2 and with the in-situ data access followed by our solution. An attachment of the entire AHN3 file repository will show how quickly metadata is extracted and inserted into the DBMS catalog's tables. Then the user will perform data imports of different areas as large as Amsterdam City Center. Once the data is imported, simple selections will be used to exemplify the export of spatial selection results into a LAS/LAZ file for offline visualization.

While executing each of the steps, the audience can visualize the query plan to understand what happens in the background and the cost of each operation. In addition, the users will have the option to create and execute queries of their own, identifying relations of their interest and obtaining further insights on the datasets. At the same time, they will be introduced to X3D viewer and its navigation commands.

3.4 Second scenario

The second scenario is composed of three parts and shows how GPU-accelerated operators contribute to a performance boost. In the first part we show how easy it is to integrate a new GPU-accelerated operator without side effects for existing operators and to consider it in the optimization strategies of a query plan.

In the second part, we will test the performance improvement achieved when the refinement step of spatial selection is done on the GPU. The execution cost will be dissected to show the absolute gains in using a GPU-accelerated refinement step instead of one using the CPU. We perform the same study in the third part, but applied to surface reconstruction.

3.5 Third scenario

The audience will have the opportunity to compare the performance of our enhanced version of MonetDB geospatial module with PostGIS boosted by a BRIN index [2]. Our aim is to demonstrate the benefits of a column-oriented architecture having *in-situ* data access and integration of spatial operators into the kernel instead of adding the desired functionality through User Defined Functions (UDF).

Using the interface illustrated in Figure 2 a 3D digital model will be requested from different backends, namely MonetDB and PostGIS. The cost of each step on each system will be presented to the user. In order to demonstrate the benefits of having geospatial functionality deeply integrated into the DBMS architecture, we will perform the triangulation step using dynamic linkage to SFCGAL [5] and by exposing it to the kernel allowing the exploitation of inline code, direct data access, and inter- and intra- operator parallelism. For performance isolation, the surface construction will be done on materialized spatial selections.

4. SUMMARY

In this work we demonstrated a column-oriented SDBMS enhanced with a set of optimized operators for managing massive point clouds. The SDBMS provides support for a flexible storage schema that allows 2D/3D geospatial datasets to store semantically rich objects that are needed for the customization (i.e., data re-generation with user defined parameters) of 3D digital city models on a large scale.

It is the first work to extend a column-oriented architecture to provide effective data skipping, efficient spatial operations, and interactive data visualization. With spatial functionality pushed down into the columnar-oriented kernel, scientists have now the opportunity to do efficient spatial analysis on data kept outside of the DBMS.

Figure 2: Web browser viewer

Such features are exploited for 3D digital city models using latest topography of The Netherlands and the latest Cadastral information for The Netherlands. Through a web-interface exploiting X3D technology, the user requests 3D digital city models with predicates on the semantic attributes.

5. ACKNOWLEDGMENTS

The work reported here is partly funded by the NLeSC project, *Big Data Analytics in the Geo-Spatial Domain* (project code: 027.013.703), and the NWO valorization project for COMMIT/, *3D geospatial data exploration for modern risk management systems*.

6. REFERENCES

[1] http://www.ahn.nl.
[2] https://www.postgresql.org/docs/9.5/static/brin-intro.html.
[3] http://en.wikipedia.org/wiki/Conceptual_schema.
[4] https://trac.osgeo.org/geos/.
[5] http://www.sfcgal.org/.
[6] http://www.kadaster.nl/web/Themas/Registraties/BGT.htm.
[7] Open Geospatial Consortium. OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 2: SQL option. OpenGIS Implementation Standard, 2010.
[8] rapidlasso GmbH. http://rapidlasso.com/, 2014.
[9] F. Alvanaki, R. Goncalves, M. Ivanova, and et al. GIS navigation boosted by column stores. *PVLDB*, 2015.
[10] R. Goncalves, M. Ivanova, F. Alvanaki, and et al. A round table for multi-disciplinary research on geospatial and climate data. *IEEE e-Science*, 2015.
[11] S. Idreos, F. Groffen, N. Nes, S. Manegold, and et al. Monetdb: Two decades of research in column-oriented database architectures. *IEEE Data Engineering Bulletin*, 2012.
[12] M. Ivanova, Y. Kargin, and et al. Data Vaults: A Database Welcome to Scientific File Repositories. SSDBM, 2013.
[13] M. Ivanova, M. Kersten, N. Nes, and R. Goncalves. An architecture for recycling intermediates in a column-store. In *SIGMOD*, 2009.
[14] O. Martinez-Rubi, M. L. Kersten, R. Goncalves, and M. Ivanova. A column-store meets the point cloud. *FOSS4G-Europe*, 2014.
[15] O. Martinez-Rubi, P. van Oosterom, R. Gonçalves, T. Tijssen, M. Ivanova, M. L. Kersten, and F. Alvanaki. Benchmarking and improving point cloud data management in monetdb. *SIGSPATIAL Special*, pages 11–18, 2015.
[16] L. Sidirourgos and M. L. Kersten. Column imprints: a secondary index structure. In *Proceedings of the ACM SIGMOD*, 2013.
[17] P. van Oosterom, O. Martinez-Rubi, and et al. Massive point cloud data management: design, implementation and execution of a point cloud benchmark. *Computer Graphics*, 2015.
[18] S. Zlatanova and et al. Towards 3d raster gis: on developing a raster engine for spatial dbms. *ISPRS WG IV/2 Workshop: Global Geospatial Information and High Resolution Global Land Cover/Land Use Mapping*, 2016.