Integrating analytics with relational databases

Raasveldt, M.

CHAPTER 1

Introduction

1 The Rise of Data Science

Analyzing data in order to uncover conclusions, often referred to as "data science", is everywhere in todays world. In order to gain value from their data, nearly every large business has a data science branch or a team of data scientists looking to extract value from their data. But data analysis is not used only in the financial sector. It is also widely used in journalism, to aid the decision of government policies, in all branches of science and in many more areas.

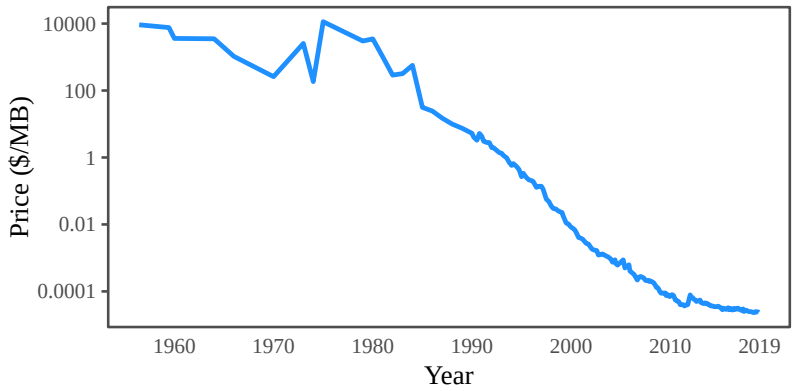

Figure 1-1: The cost of hard disks over time.

These developments are happening largely because of how cheap gathering, storing and analyzing large quantities of data have become. When we look back just sixty years, the IBM 350 disk storage unit was released. The IBM 350 could hold up to 3.75MB of data, and cost approximately 35000USD at the time. Today, we can buy a HDD with 1TB of storage for around 50USD. To put that into perspective, in 1960 the price of a Chevrolet Impala was around 3000USD. If the price of cars had fallen at the same rate as the price of hard disks, the newest model of Chevrolet would cost a mere $4.25 and could drive 200,000 times faster.

Not only has the cost of storing the data become so much cheaper, so has the cost of reading and processing that data. CPU processing speeds have improved orders of magnitude following Moore's Law, and RAM sizes have blown up. The phone in your pocket has over 10 times more high-speed RAM than the Cray-1 supercomputer had storage, and has significantly more processing power as well.

Looking at these numbers, it is no wonder that data science has become so ubiquitous. Analyzing large amounts of data has become very cheap and accessible even to small companies and individuals. Expensive supercomputers are no longer needed to store and analyze large amounts of data. Data science can be performed on cheap commodity hardware. Analyzing 10GB of data on a laptop is common place, and it is not unheard of to process 100GB or even 1TB of data on a desktop computer.

2 Tools of the Trade

While data science might appear like a brand new field, it is closer to a mixture of different fields. In particular, it is a combination of mathematics, statistics and computer science. Many of the techniques applied by data scientists are in fact techniques from the statistics field that can now cheaply be applied to large quantities of data because of technological advances.

Many of the tools that are used in data science have actually been designed and created by statisticians. An example of this is the R project for statistical computing [69]. The R language started as an open-source implementation of the

S language, a statistical language designed by John Chambers at Bell Labs. R was originally implemented by the statisticians Ross Ihaka and Robert Gentleman at Auckland for the purpose of teaching introductory statistics courses. It has grown into a tool that is used worldwide to perform statistical analysis, data classification and data visualization.

Another popular language for data science is Python, together with the support of the numeric python extensions NumPy [84], SciPy [48] and Pandas [64]. While Python itself does not have its roots in statistics, the numeric python extensions are based primarily on the APL family of languages, which includes S (the precursor of R), Fortran and MATLAB. These languages have all been designed primarily for use in numeric computing and statistics.

3 Data Science & Data Management

One of the consequences of the origins of these tools is that proper data management was never a first class citizen. Data management is largely treated as an afterthought in these tools. Typically, the data that is used for analyses is loaded from a data source into structures residing in memory and then kept around in memory. The tools do not support larger than memory data sets. Any management of that data is not handled by the tools themselves, and is left up to the user.

Data scientists typically opt to store the data in a set of flat files, as this is the most natural way of interacting with these tools. While flat file storage is simple when dealing with very small data sets that fit in individual files, it does not scale well. Flat file storage requires tremendous manual effort to maintain when the data sets grow in size. The files are also difficult to reason about because of the lack of a rigid schema, and it is difficult to share the data between multiple users. Furthermore, adding new data or modifying existing data is prone to corruption because of lack of transactional guarantees and atomic write actions provided by these tools.

All of the problems of flat file storage are not new problems. In fact, database management systems were created precisely to solve many of these problems. Modern

database management systems prevent data corruption through strong transactional guarantees and ACID properties, automatically manage data storage and make data easier to reason about by enforcing a rigid schema. In addition, the database management systems can perform efficient execution on larger-than-memory data, and allow safe concurrent access to the data.

However, despite the existence of database management systems, data scientists typically opt not to use them in conjunction with these analytical tools. This leads us to our main research problem:

Research Problem How can we facilitate efficient and painless integration of analytical tools and relational database management systems?

4 Our Contributions

In this thesis we work to answer the main research problem by investigating the different methods of combining relational database management systems and analytical tools. We consider the three separate methods of connecting analytical tools with RDBMSs: (1) client-server connections, (2) in-database processing and (3) embedded databases. For each of these methods, we examine the current state of the art and attempt to improve on it in both run-time efficiency and usability.

- **Client-Server Connections (Chapter 3).** We examine the client-server protocols of popular RDBMSs, and evaluate their effciency in the context of large-scale result export that is required to perform data analysis and machine learning on large data sets contained within these systems. Based on this analysis, we propose a new client-server protocol that handles these situations more efficiently and show its efficiency by implementing it in two open source RDBMSs.

- **In-Database Processing (Chapters 4 and 5).** We examine current methods of in-database processing in popular RDBMSs and improve on these methods by implementing a new method of in-database processing aimed at accelerating

in-database analytics: Vectorized UDFs. We implement these in MonetDB, a popular open-source RDBMS, and show how these UDFs can be effectively used to perform analytical workflows entirely within the RDBMS.

- **Embedded Database: MonetDBLite (Chapter 6).** We adopt the popular open-source RDBMS MonetDB to run as an embedded database inside analytical tools. We show how an embedded database can greatly increase usability of a database system, as well as show how the speed at which the analytical tool and the RDBMS can exchange data is greatly improved by embedding the database.
- **Embedded Database: DuckDB (Chapter 7).** Learning from our implementation of MonetDBlite, we identified the requirements and challenges of an embedded database system, and created our own RDBMS designed for being embedded from scratch: DuckDB. DuckDB fixes many of the deficiencies of MonetDBLite that were caused by the system being initially designed as a stand-alone server process.

5 Structure and Covered Publications

We present the background material necessary to understand this thesis in Chapter 2. We discuss the history of relational database management systems, and how they relate to the field of analytics, and we discuss the various ways in which database systems can be combined with stand-alone analytical tools.

In the subsequent chapters, we discuss the methods in which we aim to improve over the existing work. In Chapter 3, we describe our work on improving the client-server protocol, based on the following paper:

- **Don't Hold My Data Hostage - A Case For Client Protocol Redesign**
  Mark Raasveldt, Hannes Mühleisen
  43rd International Conference on Very Large Data Bases (VLDB 2017)

In Chapter 4 we discuss our work on extending user-defined functions for analytical use cases. This chapter is based on the following paper:

- **Vectorized UDFs in Column-Stores**
  Mark Raasveldt, Hannes Mühleisen
  28th International Conference on Scientific and Statistical Database Management (SSDBM 2016)

In Chapter 5 we discuss our work on embedding analytical workflows inside a database system. This chapter is based on the following paper:

- **Deep Integration of Machine Learning Into Column Stores**
  Mark Raasveldt, Pedro Holanda, Hannes Mühleisen and Stefan Manegold
  21st International Conference on Extending Database Technology (EDBT 2018)

In Chapter 6 we discuss our work on extending the MonetDB system into an embedded database system called MonetDBLite. This chapter is based on the (currently unpublished) paper:

- **MonetDBLite: An Embedded Analytical Database**
  Mark Raasveldt and Hannes Mühleisen

In Chapter 7 we discuss our work on creating the embedded database system DuckDB. This chapter is based on the following paper:

- **DuckDB: an Embeddable Analytical Database**
  Mark Raasveldt and Hannes Mühleisen
  ACM International Conference on Management of Data (SIGMOD 2019)
  Demonstration Track