

Enabling Agile BI with a Compressed Flat Files Architecture

William Sunna and Pankaj Agrawal



Dr. William Sunna is a principal consultant with Compact Solutions.
william.sunna@compactsolutionsllc.com



Pankaj Agrawal is CTO of Compact Solutions.
pankaj.agrawal@compactsolutionsllc.com

Abstract

As data volumes explode and business needs continually change in large organizations, the need for agile business intelligence (BI) becomes crucial. Furthermore, business analysts often need to perform studies on granular data for strategic and tactical decision making such as risk or fraud analysis and pricing analysis. Rapid results in active data warehousing become vital in order for organizations to better manage and make optimal use of their data. All of this triggers the need for new approaches to data warehousing that can support both agility and access to granular data.

This article presents a new approach to agile BI: the compressed flat files (CFF) architecture, a file-based analytics solution in which large amounts of core enterprise transactional data are stored in compressed flat files instead of an RDBMS. The data is accessed via a metadata-driven, high-performance query engine built using a standard ETL or software tool.

When compared to traditional solutions, the CFF architecture is substantially faster and less costly to build thanks to its simplicity. It does not use any commercial database management systems; is quick and easy to maintain and update (making it highly agile); and could potentially become the single version of truth in an organization and therefore act as an authoritative data source for downstream applications.

Introduction

Large enterprises often find themselves unable to use their core data effectively to perform BI. This is mainly due to a lack of agility in their information systems and the delays required to update their data warehouses with new information. As business climates change rapidly, new dimensions, key performance indicators, and derived facts need to be added quickly to the data warehouse so the

business can stay competitive. In addition, access to historical, low-granularity transaction data is vital for tactical and strategic decision making.

Traditional data warehouse solutions that use relational databases and implement complicated models may not be sufficient to satisfy the agility needs of such BI environments. Introducing new data into a warehouse often involves relatively long development and testing cycles. Furthermore, the traditional data warehouse architectures do not adequately cope with many years of transactional data while meeting the performance expectations of end users. Enterprises often settle for summarized data in the warehouse, but this severely compromises their ability to perform advanced analytics that require access to vast amounts of low-level transactional data.

With all of these inconveniences, the need for an agile solution that can handle these challenges has become acute. This article presents an innovative architecture that

As business climates change rapidly, new dimensions, key performance indicators, and derived facts need to be added quickly to the data warehouse so the business can stay competitive.

offers a cost-effective solution to create large transactional repositories to support complex data analytics and has agile development and maintenance phases.

In this architecture, the core enterprise data is extracted from operational sources and stored in a denormalized form on a more granular level in compressed flat files. The data is then extracted using a high-performance extraction engine that performs SQL-like queries including selection, filtering, aggregation, and join operations. Power users

can extract transactional or aggregated data using a simple graphical interface. More casual business users can use a standard OLAP tool to access data from the compressed flat files.

The benefits of CFF architecture are manifold. The infrastructure cost of a CFF solution is substantially lower, as RDBMS license costs are eliminated. Storing data in standard compressed flat files can reduce disk storage requirements by an order of magnitude. This not only reduces cost, but also allows an organization to provide many more years of transactional data to its analysts, allowing for much richer analysis. In addition, the simplified architecture can be built and supported by a much smaller team. This article will explain the CFF architecture through a simple case study; discuss the metadata-driven feature of the architecture; and compare the CFF architecture to traditional architecture, with an emphasis on agility.

Case Study

We will use a simple case study to demonstrate the CFF architecture. Suppose researchers and pricing analysts in a major retail chain want to study the sales trends and profitability of the products sold at their stores located in all 50 states. To support their analyses, they need 10 years of detailed sales transaction data available online.

Let's assume the chain sells more than 30 categories of products such as automotive and hardware. Each category contains a wide range of products. For example, the automotive category contains engine oil, windshield washer fluid, and wiper blades; each of these products has a unique product code. Once a day, all the stores send a flat file containing point-of-sale (POS) transactions to headquarters. In addition to product and geographical information, the transactions also contain other information such as the manufacturer code, sales channel, cost of the product, and sale price.

Assume that most users' analysis is based on the geographic location, product category, and the accounting month in which the products are sold. Let's refer to such attributes as "major key attributes." For example, a business analyst may request a profitability report for a selected number of products in a given category in Illinois in the first quarter of 2009.

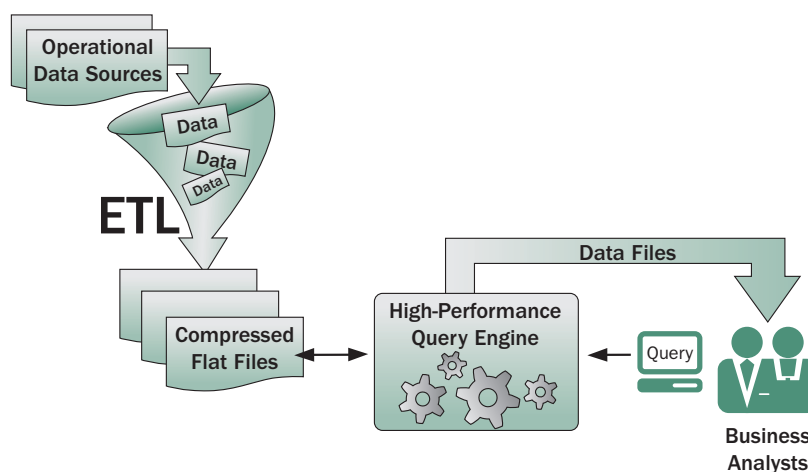


Figure 1. Overview of CFF architecture

The Compressed Flat Files Architecture

The CFF architecture (Figure 1) is best characterized by its simplicity, yet it delivers many invaluable benefits. The architecture is highly metadata-driven to allow flexibility and agility in development and maintenance. The architecture also allows the enterprise to implement a security layer to regulate data access. This section describes how the data is generated, organized, and extracted, along with the CFF metadata-driven characteristics.

Data Generation and Organization

In this step, the data is extracted from operational data sources using a standard ETL tool. The data can be extracted from legacy systems, operational databases, flat files, or any other available data sources. The extracted data can be on a very granular level, such as POS transactions for a certain retail chain, as described in the case study. For the widest possible use, we recommend storing the data at the most granular level. The data is then cleansed and transformed according to a set of business rules, then partitioned, compressed, and stored in multiple files on hard disk. A set of key performance indicators (KPIs) is also calculated at this stage, again at the most granular level.

The way the data is organized and distributed in compressed flat files is a key factor in the success of this architecture. Similar to commercial database partition

elimination mechanisms, the compressed flat files should be organized to optimize extraction as much as possible. In other words, the main goal is to read as few files as possible to satisfy any given query. To ensure this is the case, some extraction patterns should be analyzed before finalizing the organization of the files.

For our case study, it makes sense to split the data files by their major key attributes. For example, we can split the files by product category, state, and accounting month because these three attributes are used in almost all the extractions. If we are storing 10 years of data for 50 states and 30 product categories, then the number of compressed flat files will be $10 \text{ years} \times 12 \text{ months} \times 50 \text{ states} \times 30 \text{ categories} = 180,000 \text{ files}$. Each compressed file should be named in a way that describes its contents. For example, given a compressed flat file, a user should be able to identify what product categories it contains for what state and what accounting month. If the file names do not describe the major key attributes, then there should be a mapping file to link the file name to its major key attributes.

Data Extraction

The extraction process starts with the end users, who compose their requests in a simple, standard user interface that can be developed using Java or .NET. The user interface should allow users to specify the data attributes they would like to see and what measures or metrics they

would like to calculate. In addition, it should provide filters to further refine the data.

Let's take a data extraction request for our case study: A user wishes to perform a profitability analysis for four products (with codes 01, 02, 03, and 04) in the automotive category, which has a code of 01, for the state of Illinois (IL) in the first quarter of 2009. The user interface allows the user to select attributes (category code, state, product code, transaction date, number of items sold, sales amount, and cost amount) and specify the relevant filters, as shown in Figure 2.

Once the user submits the request, the query details are passed to the high-performance query engine that is responsible for extracting data directly from the compressed flat files. The query engine will first build a list of the compressed flat files needed for the extraction based on the major key attributes selected. In our example, only one category code has been requested for one state during a three-month period. Therefore, only three compressed flat files out of the 180,000 total files are needed to satisfy the request. This early selection of files represents a huge

up-front performance gain in query processing and is one of the major strengths of the CFF architecture.

The query engine then reads the data in the relevant files and applies additional data filters such as the product code. The next step will be aggregating the measures requested (sales amount and cost) by product code and presenting the results to the analyst. The resulting data sets can be produced in any format, such as comma-separated or SAS-formatted files. Note that the user interface presented here is to be used as a data extraction interface, as opposed to a standard reporting or presentation interface. Standard BI tools such as MicroStrategy and Business Objects are also supported by this architecture.

The high-performance query engine can be implemented with any commercial or open source ETL tool, or it can be built using any programming language. If the organization uses such tools and software, then there will be no need to purchase additional licenses for a database management system.

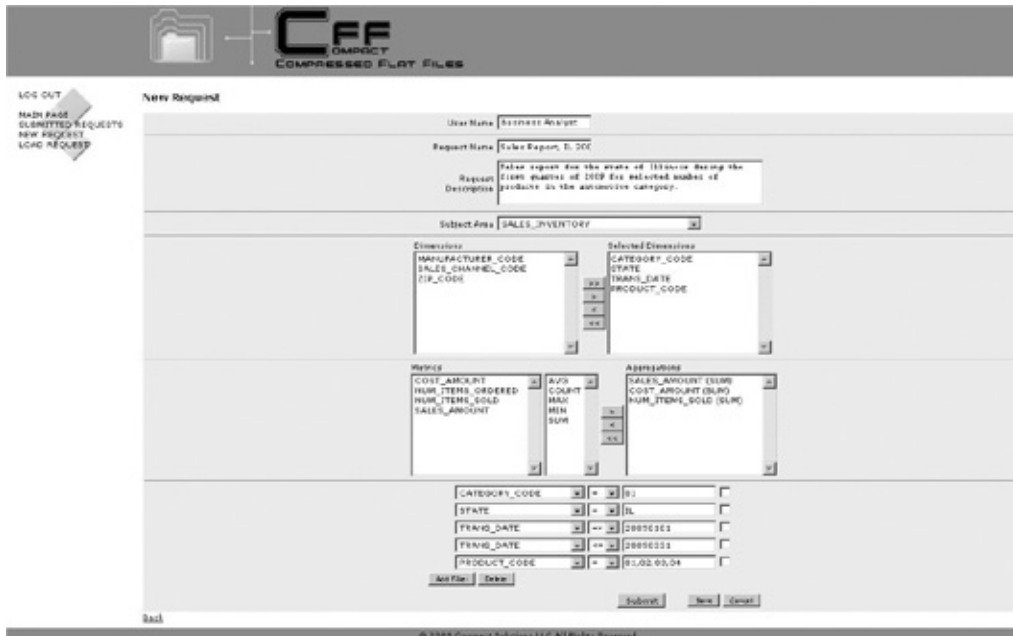


Figure 2. Example of a query user interface

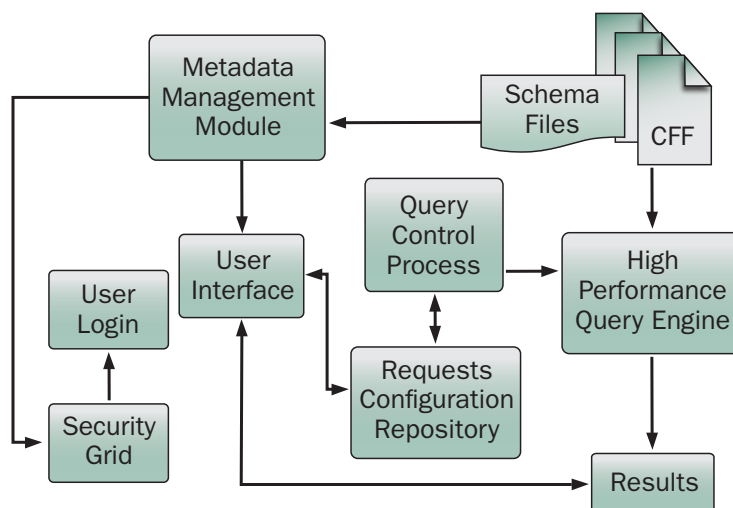


Figure 3. Metadata-driven architecture

Metadata-driven Approach

The CFF architecture is highly metadata-driven to allow for maximum agility in both the initial build of the application and any required maintenance in the future. Due to the simplicity of the data model manifested in the CFF, the data layouts (schema files) of the CFF are leveraged to generate the contents of the user interface via the metadata management module, as shown in Figure 3. Therefore, the addition of new fields or modifications to existing fields are reflected in the user interface unit without requiring any programming effort.

The metadata management module also takes into consideration the classification of attributes in the data as specified in the schema files; it distinguishes major key attributes from other dimensional attributes and measures. Furthermore, it provides user privileges information to the interface by consulting the security grid module, which contains privileges and security rules for data access. The user interface builds custom data extraction menus for different users depending on what they are allowed to query or extract.

All user requests are deposited in the requests configuration repository, a standard, secure location that contains the specifics of each request. This allows users to access any requests they submitted in the past, modify them if needed, and resubmit them. The query control process gathers new requests from the request configuration repository and submits them to the high-performance query engine. Queuing of requests, priorities, and other scheduling considerations are implemented in the query control process.

Traditional Architectures, CFF, and Agility

The CFF presents an alternate way to implement complex data analytics solutions with huge gains. Compared to traditional architectures, it is significantly faster to build due to its simplicity. It is far easier to maintain due to its metadata-driven characteristics. Systems based on this architecture can provide very rich information to analysts because very large amounts of highly granular data can be kept online at a fraction of the cost of traditional architectures. In one implementation of this architecture, more than 100 power users at a large insurance company perform complex analytics on 22 years' worth of claims and premium transactions.

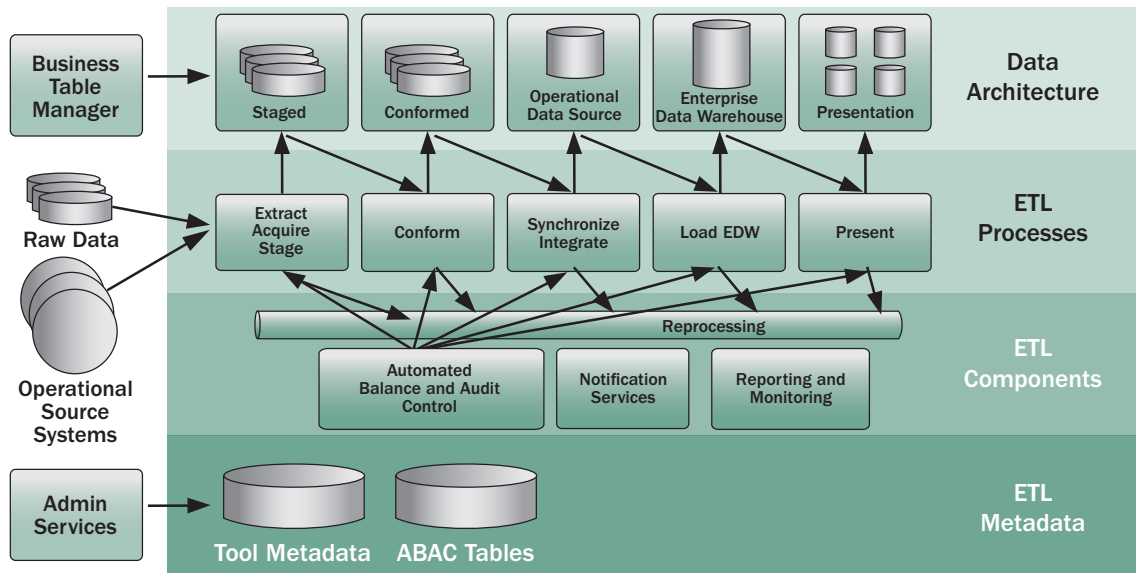


Figure 4. Traditional data warehouse architecture

Traditional data warehousing solutions based on relational databases require many layers of data models with corresponding ETL processes, making the architecture very complex, as shown in Figure 4. The traditional data architectures usually require separate models to be built for staged data, conformed data, the operational data store, a data warehouse or data mart, and presentation layers. These models are populated by multiple ETL processes.

Because this architecture depends heavily on an RDBMS for storing data, data is often aggregated to provide better performance and manage data growth. Because of the very large data volumes involved, it is extremely expensive to store many years of transactional data in such data warehouses. Therefore, most such solutions keep a small amount of granular data (say a few months) in base tables, and rely heavily on aggregated data to meet user demands. Such aggregated data is often of limited use for applications such as risk and fraud analysis, price modeling, and other analytics that require a longer historical perspective.

If we compare the CFF solution to a traditional data warehousing solution on basic development and maintenance activities, we can easily recognize the agility gains offered by the CFF architecture. Table 1 compares the

CFF architecture with the traditional architecture along some key criteria.

During the development phase of the CFF architecture, adding new attributes or deleting/updating existing attributes requires making changes to only one repository and one ETL application, whereas the traditional architecture requires changes in many places. In fact, this simple difference can save substantial time, money, and resources because it eliminates the need to build many sophisticated models, whether dimensional or normalized in a relational database. Since data is stored in only one repository (the set of compressed flat files), only one set of ETL routines needs to be developed, saving time and money. Thus, the architecture is intrinsically agile.

During the maintenance phase, inserting new attributes in the data is easier in CFF because of its metadata-driven nature. Once the CFF layout is modified, the rest of the updates are done automatically all the way to the user interface. In a traditional solution, the new attributes have to be propagated from one process to another and from one data model to another, requiring significant development and testing. Making a small change requires the involvement of data modelers, database administrators, ETL developers, and testers.

Phase	Change Step	Traditional Architecture	CFF
Development	New attribute	Updates to several layouts, data models, and ETL processes	Updates to only one layout and one ETL process
	Delete attribute	Updates to several layouts, data models, and ETL processes	Updates to only one layout and one ETL process
	Update attributes	Updates to several ETL processes	Updates to only one ETL process
Maintenance	Insert attributes	NULL for historical data and layouts; updates to several ETL processes going forward	Easier with metadata automation; updates to only one ETL process
	Delete attributes	Nullify column; updates to several ETL processes going forward	Easier with metadata automation; updates to only one ETL process going forward
	Update attributes	Updates to several ETL processes	Updates to only one ETL process

Table 1. Comparison of CFF solution and a traditional data warehouse solution

Summary

According to Forrester Research principal analyst Boris Evelson, the slightest change in a traditional data warehouse solution can trigger massive amounts of work involving changing multiple ETL routines, operational data store attributes, facts, dimensions, major key performance indicators, filters, reports, cubes, and dashboards. Such changes cost time and money. This frustrates IT managers and business users alike. The need for agile data management has, therefore, become acute. Such solutions should not be driven by what tools are available but by smart strategies and architectures.

In response to business needs for agility and lower cost, we have presented a new but proven data management architecture, the compressed flat files architecture. We have demonstrated the simplicity of this architecture and how it can be used to satisfy business needs in an agile environment. We have shown how this architecture is independent of any technologies or tools. We also demonstrated how it allows business users to analyze vast amounts of data at the most granular level without any loss of detail, a feature that would be prohibitively expensive to build using a traditional solution.

We compared the CFF architecture with traditional architectures to demonstrate the agility of CFF in multiple activities in the development and maintenance phases. We have shown that the CFF architecture offers important benefits: reduced development time due to simplicity and metadata-driven architecture; reduced cost from eliminating the need to use a relational database management system; and the ability to store much larger amounts of data on smaller storage devices.

A solution based on the CFF architecture has already proved its value at a large corporation where it handles more than 50 TB of raw historical transactional data. Furthermore, the CFF architecture has been recognized by data warehousing experts such as Bill Inmon and industry analysts such as Forrester as an important evolutionary step in data management and BI.

Today's BI challenges require non-traditional solutions to rein in the cost and complexity of managing data, as well as more agile responses to business changes. The CFF architecture meets these requirements. ■