3

Architectures for embedded analytics

Having looked at the nature of decisions and a design pattern for analytics, how do we start to put together an embedded analytics solution? In this chapter we will take an overview of some architectural features to consider. We will also look at some benefits and weaknesses of common approaches.

# Elements of embedded analytics

When thinking about how to approach embedding analytics in your applications, look for several common elements.

## Data connectivity

We will consider the topic of data for embedded analytics in much more detail in Chapter 4. But for now, we should recognize and understand that simply being able to connect your data source is a critical capability of any analytics solution. This is not always as obvious as it may sound.

When we embed analytics into operational systems, we often connect to legacy applications which the business has been running for years. Or perhaps we need to access data in custom applications. And in both these cases we may have to integrate both operational and analytic data. If you cannot connect, all these scenarios will fall at the first hurdle.

## The analytic engine

Analytics is more advanced than reporting because the pattern of Orientation – Glimpsing – Examining – Deciding is predicated on interactivity, especially Examining. Often, when we have glimpsed a potential insight and examined it, we discover things are not quite as we thought and we return to glimpse another, related insight to examine again. This process can prove quite convoluted. Think of the steps you go through searching for a restaurant or hotel online – internet search engines employ our analytic pattern too!

An embedded report (see below) has limited scope for this kind of interaction. You may filter or sort the data in the report, but just in the same way as a regular report might be filtered or sorted. There is nothing especially “analytical” or insightful about an embedded report. The Report Server may need to scale substantially (see below), but they rarely have complex calculations to perform. Similarly, a visual component added in code may be used only to display results, and not have any real analytics behind it.

Embedded analytics, on the other hand, may include quite sophisticated options for selections, navigations and even custom queries or calculations. The visualizations may display data in complex and demanding ways, perhaps enabling us to zoom in from a summary to fine detail. In such cases, the analytic engine is critical.

What is the analytic engine? Generally speaking it’s a system for performing queries against a data source which includes specific functions for analyzing data. An OLAP system (the acronym stands for On-Line Analytic Processing) can typically perform numerous aggregations such as sums, averages, minimum and maximum values across large volumes of data with near-instantaneous performance. And of course, the aggregations may be much more complex than these examples. Other analytic engines may have particular knowledge about time-series of data and be able to perform calculations such as Year-To-Date or Same-Period-Last-Year with a simple query for developers and great performance for users. More recently we see analytic engines with specific features for machine learning algorithms and data science.

Expect an analytic engine with high performance, preferably one with query optimization to take best advantage of available resources and the best methods of computing results.

## Branding the user experience

It is a common requirement that an embedded solution should not look like an embedded solution! Users want the analytics to look completely consistent with the host application. This expectation may require that navigation, selecting, editing, confirming changes and so on should all behave the same way in the embedded environment as in the host.

Moreover, the host environment may be branded with logos, corporate standard colours and preferred typefaces. When we white label the user experience, all these elements will look the same in the embedded analytic objects. This is a very common requirement, although I am personally skeptical of its value. On the other hand, I understand the resistance to the other extreme, where the analytics vendors name, logos and UX preferences are all visible in what should be an unintrusive embedded component.

Seamless is not what it seems

There’s a phrase you will read quite often when you research user experiences or any architectural integration. It is seamless.

Seamlessness, we are told, is a virtue: the idea being that when we switch from one context to another in our work, the cognitive effort required reduces our productivity and distracts us. The solution is an environment where we do not need to switch context but see all our different scenarios working together as a unified whole – a seamless garment.

Metaphors can be tricky. Seams are, in fact, very important. They do not merely hold together a garment stitched from smaller pieces of material. Seams give shape to our clothing, so that it drapes properly and moves with us. Seams give strength. Seams are very practical.

So too in our software design. Sometimes a user will not want to switch context – they just need a quick reminder of a number, or to glimpse a trend. But at other times, we absolutely do want our user to switch context: they should be aware that now I am doing analytics and this involves a slightly different practice from my regular work.

Think carefully, therefore, about seamless experiences. What does the user need to know about what they are doing? Will a clear transition from one mode of thinking to another help them? Sometimes, you need a seam.

## Developer resources

In this chapter we review several different criteria which may influence your choice of embedded analytics architecture. However, in many cases I have seen the decision come down to just two simplified, if rather extreme, parameters. Can I connect to my data? That is essential, of course. But also, can my developers do this work effectively? Have they the skills, the resources and the time?

In general, we may prefer an embedded architecture which reuses as much of our existing effort as possible, whether in data, analytics, security or application management

## Scalability

When embedding analytics in operational applications you should be aware of two different forms of scalability.

Firstly – and most familiar – is the volume of data which the analytic system can handle. When vendors talk about large-scale solutions this is generally what they mean. At the high end, modern engines are designed to handle even Petabytes of data, but because the design of analytics engines varies widely, it is useful to ask if your candidate system can handle all the data it needs to? This is unlikely to be a problem if you are integrated with, say, an HR system or a customer record system: you will almost certainly not have enough employees or customers to cause any issues with even a basic reporting system. However, if connecting to an enterprise data lake and summarizing Big Data, such as web logs, ecommerce transactions, or Internet of Things device logs, then scale really can prove problematic, especially for in-memory analytics engines which must hold all the data at once to work with it effectively.

A second form of scalability is the number of users who may connect to the system, especially simultaneously. This varies throughout the day and the week - see the sidebar, The 9am problem – but there may also be seasonal trends depending on your business.

The 9am problem

I first started working with commercial enterprise servers in the 1990s. Previously my work had focused on complex but smaller scenarios, often in research. But when designing reporting and analytic applications for large clients such as oil companies, retailers and healthcare providers I became very familiar with a problem of scale. At 9am each weekday, and especially on Mondays, almost every manager would log into the system and refresh their critical reports. This was the peak load on the system and we quickly learned that simply throwing resources at the problem – such as larger servers – was not a popular solution, mainly because it was so expensive especially when, by 9:30, the demand was over. We used a variety of techniques, including predictive pre-processing, caching, cache-sharing and just-in-time processing to make it work.

You may not have the same problem today. In fact, in these days of cloud computing you should not. Your cloud service provider should be able to provision truly elastic scalability which is both efficient and cost-effective. However, you should take care to model this scenario – calculating what the peak demand will look like – and ensure that it is indeed reliably implemented by your cloud provider and your analytics engine.

## Security

In Chapter 6 we will dig deeper into the security implications of embedded analytics and in Chapter 7 we will also look at some topics relating to data governance. Here, it is important to point out some key criteria to consider when choosing a platform:

* Ensure the analytics system can implement the right protocols for your needs. Look for standard integrations such as LDAP, oAuth or Active Directory.
* Role-based security is critical. You must be able to configure permissions to see and work with data based on the role of the user. Even better, you should inherit these permissions directly from the hosting application, so you do not have to re-invent the wheel and define the same security requirements twice.
* Single sign-on (SSO) could be essential for you. If a user has to sign in to their host application and then sign in again to their embedded analytics, it is a clumsy and error-prone process. SSO enables just one login for access to all the related applications.

## Administration tools

When you must manage all these elements – data access, security, scaleability and the user experience – you will appreciate effective administrative tools.

IT are people too! In all our talk of the user experience and the need for effective business user tools, it is easy to overlook the needs of IT. But let’s remember that IT teams work day-in-day-out administering servers and applications. They will also suffer from context switching if user experiences are badly designed. Moreover, if they are fatigued, stressed or just working with clumsy and inefficient tools, their mistakes can have consequences that extend far into the enterprise. So please, consider your IT staff: they deserve – and they need – excellent applications too.

One important administrative feature is the scheduling of updates and data refreshes. Particularly when working with a host application which also refreshes on a schedule, be sure that you can synchronize, share or re-use schedules as needed.

# Embedded analytics platforms

Having described some of the elements of an embedded platform, we can now look at different approaches to embedding, each with their own benefits and weaknesses. We will consider component libraries for developers, Enterprise Reporting and Business Intelligence platforms which can also be used for embedding and finally, purpose-built platforms specifically geared for embedding.

## Component libraries

It can be very tempting for developers to “roll their own” solution – building the entire embedded analytics infrastructure from scratch. The temptation arises because component libraries for visualization are very attractive, offering a wide range of chart and graph types from the most basic to the most sophisticated.

Perhaps your needs are simple: your budget is restricted; your configuration is static and you don’t expect to integrate more demanding analytics in the future. In such cases, you can certainly look at component libraires as an effective way to get started.

Equally, if you have a specific need for a specialized and sophisticated chart type, then a visual component library such as d3.js (<https://d3js.org>) or Highcharts (<https://www.highcharts.com>) may be your best, or indeed only, option. See Figure 3-1.

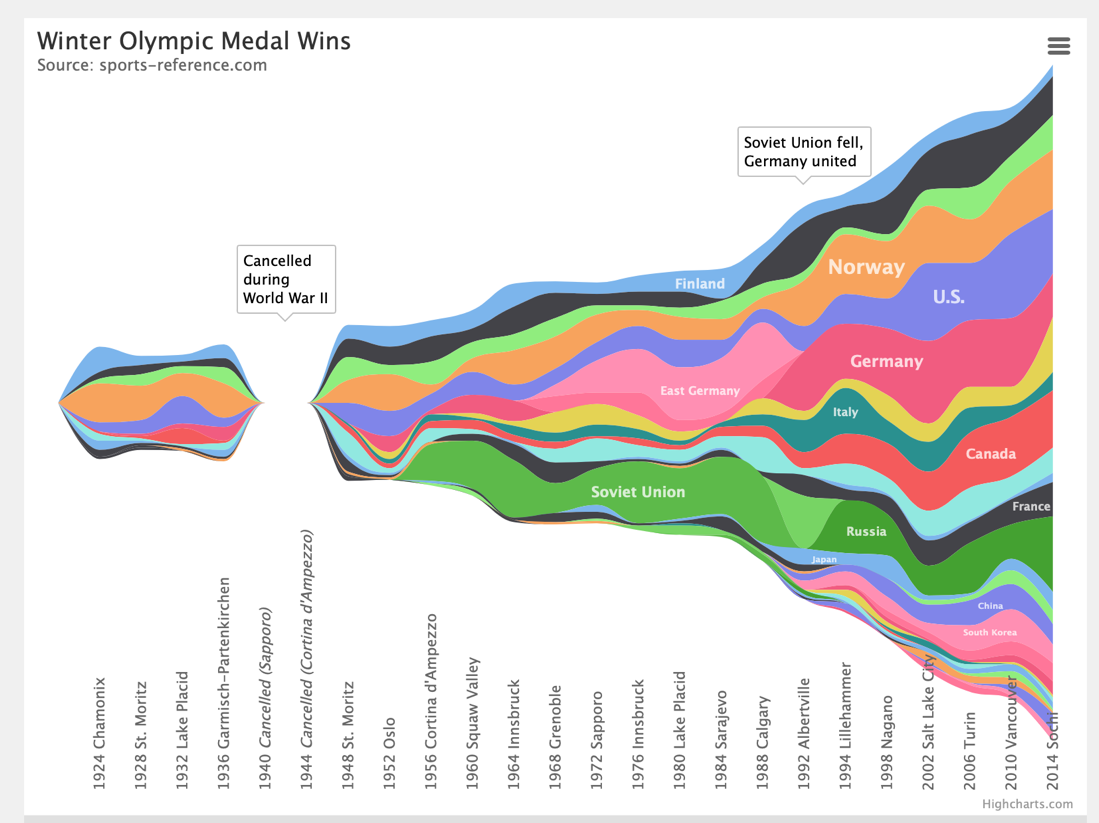


Figure -. A sophisticated chart, built with the Highcharts visualization library.

However, there are significant drawbacks to visualization component libraries as a practical solution for businesses focused on operational and tactical decision-making. Development can prove time-consuming because every component must be customized in code, with security and administration also defined in code at a component level. While this may work as a quick way to get started, maintenance becomes a real problem when, for example, data connections must be redirected to a new source, or the application is rolled out to a wider team requiring different security configurations.

Another concern for development teams arises when building analytics redirects resources from other development tasks. If those developers are not already specialists in data and analytics, their learning curve can be steep. And the problems of administration and security remain.

I once advised a computer gaming company who had an excellent, highly skilled development team. In fact, almost all the senior people were practiced developers who had arrived in their roles along separate paths but with a software background. They were also visual thinkers who enjoyed working with sophisticated charts and visualizations. As a result, many of the managers, whether in marketing or product management had built their own visual tools, connected to the corporate data lake, using component libraries. Yet even for these sophisticated users, governing data access and security, managing versions and applying corporate role-based rules was too much to do in code. So rather than embedding analytics in operational applications, they chose a Business Intelligence which could be extended with custom components. The BI platform included features for security, governance and administration – the component library offered the visualizations they wanted. This rather topsy-turvy way of doing things worked for them, but only because everyone from the CFO to the secretary was a developer at heart. I don’t recommend it, but the story is worth telling if only to remind us that there’s a solution for every scenario, if we are innovative enough.

In short, although component libraries can offer very sophisticated visual capabilities, the burden of maintaining connectivity and data management in code is likely to be too much effort in deployment for all but the simplest scenarios.

## Enterprise reporting platforms

If you work in a large organization, you almost certainly have an existing enterprise reporting platform and this may be the easiest place to start with embedded analytics. These applications are designed to provide a standardized view of business data at scale across your business. Typically, a centralized IT team develops, maintains and deploys the platform and applications, which helps with standardization but limits customization and agility. In fact, development and deployment cycles can end up quite extended.

Nevertheless, reports can embed quite easily into an application, typically populating an inline frame in HTML; all we must do is point to the report’s address on an enterprise server. However, simplicity often comes with limited functionality. Interactions with the report are often limited to sorting, searching and some simple filtering. Equally, embedding can raise all sorts of issues for reporting platforms. Reports with large numbers of records may not to paginate properly within a frame, and scrolling to find records does not help with our Orientation – Glimpsing – Examining – Deciding model from Chapter 2.

Some features of enterprise reports may prove useful in an embedded scenario. For example, when data repeats in groups, reports can be banded as in Figure 3-2. Be careful, however, that these experiences, intended for reports which stand alone, are appropriate and not distracting when embedded.



Figure -. A banded report

Good scenarios for reports include where you need to embed a standardized enterprise view in a simple format, where limited interaction is acceptable and where data – or at least summary fields – are easily presented within a frame. Do ensure that the report is genuinely relevant to the workflow of the hosting application. Some scenarios, however, do not have an operational workflow as such. For example, reports can be embedded in an enterprise portal such as Microsoft SharePoint as in Figure 3-3. These reports are largely static objects: perhaps we should not think of this use case as truly analytic.

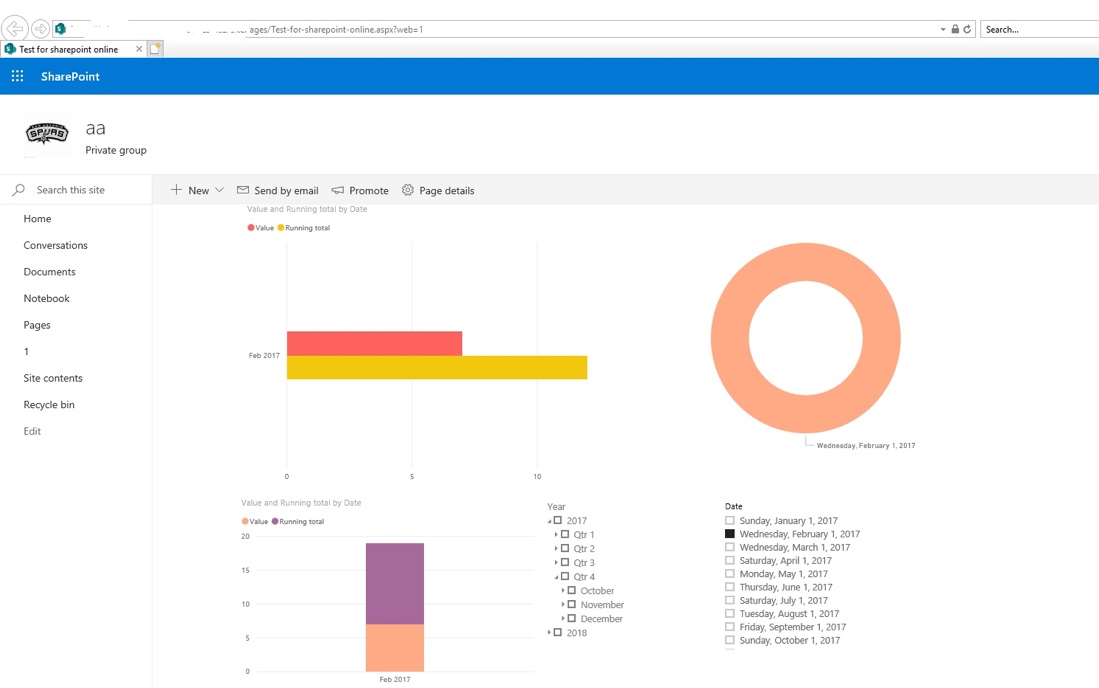


Figure -. A report embedded in a Microsoft Sharepoint enterprise portal page

Embedding a report that does not provide the most relevant information in the right context will be at best a waste of effort and too often confusing, clumsy and counter-productive. And reports, even when embedded, look like reports. They are mostly for reference and do not open you to new perspectives or discoveries.

## Business Intelligence applications

When we move beyond the simplest reporting and dashboard solutions, Business Intelligence applications offer more in the way of customization and the ability for users to perform their own exploration and analysis. These BI platforms often include good navigation features for Orientation, a wide range of visualizations for our Glimpsing, and powerful interactive features for Examining data. However, they do require more training than simple reports to be used effectively. Several leading BI platforms use in-memory processing which promises great performance for even complex scenarios but may limit the use of big data in data lakes.

Having said these applications offer more interactivity than reports, I should also point out that this flexibility can come with its own costs. The user experience – especially the navigation or selection features – may be different from your host application. Where you have one method of moving between records, or selecting records, in the host application and another in the embedded analysis, can be confusing and distracting.

You may also find the BI application has its own security model and security architecture with the result that it may not be able to inherit permissions from the host application.

However, when the BI application is already widely deployed and used in your business, there’s a real convenience in embedding this into a host application. In such cases, users are not learning a completely new methodology. Indeed, they may appreciate a familiar environment.

While some BI applications can be embedded into a host application as easily as a report – using a frame, for example – several have more advanced features for embeddability, such as dedicated container components and APIs for developers. Using an iFrame is very attractive if you simply want to add well-defined sections of analytic features, or even specific analytic tabs to your existing application.

Although developers may appreciate the APIs included with Business Intelligence applications, too often they find the programmable or embeddable functionality limited and clumsy to work with. That can happen when the vendor develops the APIs some time after the BI application itself: in other words, when they are added-on rather than built-in. APIs developed in this way are likely to be less well integrated, less reliable (being more complex to test) and less carefully defined. These APIs may also be less fully documented.

For many businesses it makes sense to re-use the work they have already put into their existing Business Intelligence development. Very often what we want to embed is not a completely new report or visualization, but a version of something which is already in use elsewhere in the organization. But as we have seen there are some limitation and compromises in this approach.

When we need to develop new solutions from scratch or when the limitations get in the way of an optimal user experience for the host application, then it is time to look at specialized tools.

## Embedded analytics platforms

When you need to develop best-in-class embedded analytics, you will naturally look for a best-in-class platform and toolset for development.

Some vendors have created platforms designed from the ground-up to be embedded and extended by developers. The APIs are an integral component of the platform architecture. Indeed, in the ideal case, the vendors own developers build the product with the same APIs external developers can use. This thorough approach avoids nearly all the difficulties of add-on APIs. At their best these APIs encapsulate the most common and most compelling functions of the system in well-documented, clear constructs which can be called simply from your own code.

Being designed from the beginning to be embedded, these platforms are also more likely to have navigation and exploration features that can be called from the hosting application with a consistent user experience. In Figure 3-4 you can see analytics fully embedded in an operational application. Indeed, it can be difficult here to tell which elements of the UX are analytic and which operational.

A good embedded analytics platform should also ship with a range of out-of-the box functions for common scenarios. This will ensure that development and deployment are fast and reliable.

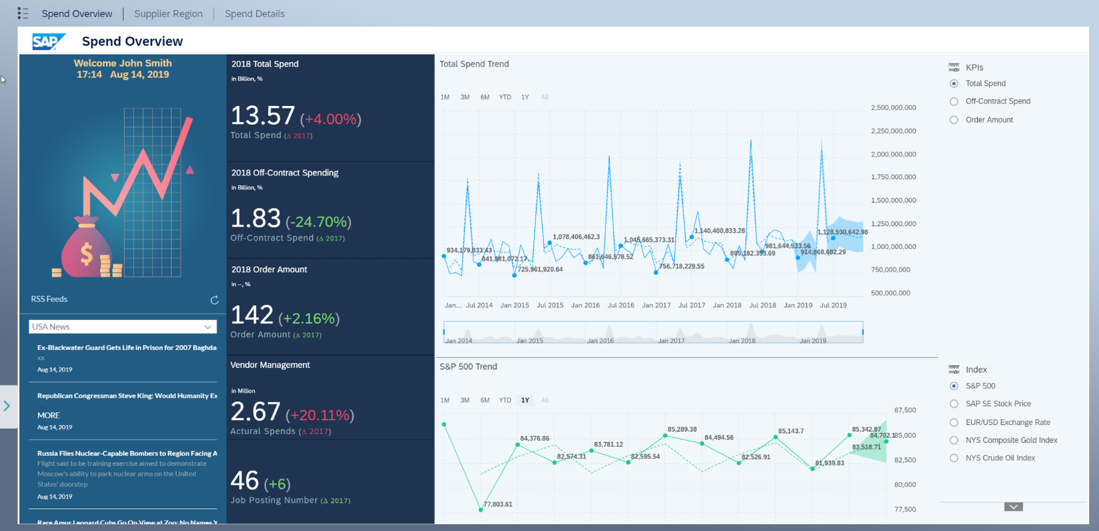


Figure -. Analytics fully embedded in an operational application.

It is important for an embedded solution to integrate with the security features of the host application. We will discuss this more in Chapter 6, but for now be aware that with a good embedded platform security integration should be straightforward.

For sure, these embedded platforms require more skills and training than a simple component library. But the completeness of their features and the power of their analytics are often worth the extra effort. These platforms should also ship with an SDK – a Software Development Kit which should include detailed documentations, sample code and components (such as JavaScript components) for data connectivity and custom visualizations which help developers to quickly build and test an application.

### Embedded self-service

Self-service analytics platforms are now the most popular form of business intelligence. This approach offers business users the chance to build their own analytics with tools specifically designed for the purpose. In theory, we can then free up IT resources from dashboard development for other work, thus removing bottlenecks. Also in theory, business users get exactly what they want without a lengthy cycle of requirements gathering and prototyping with an IT team.

In practice, self-service rarely works out quite so ideally. As business users create more analytic artefacts (reports, dashboards and low-code applications) so they demand more access to data, which IT must at least govern and, in most cases, must provision. Also, as these analytics proliferate system performance can degrade (business users rarely build efficient applications) and problems of availability, governance and compliance can increase. In a self-service system, IT either have to run to keep up with business users, or rethink their role. See the sidebar “From gatekeepers to shopkeepers” in Chapter 7 for more thoughts on this dilemma.

You might expect that self-service is not practical in an embedded scenario. However, if your platform of choice has these capabilities, users can avoid a wide range of requests to IT with some simple customization. Perhaps all we need is the ability to change a visualization type. (In spite of the best efforts of designers and visualization experts, some people still prefer a pie chart!) But you may also add new filters, new selections or even completely new reports based on a current view.

Most embedded self-service scenarios, in contrast to stand-alone self-service platforms, do not expect business users to build everything from scratch. Instead, business users may reuse existing objects which the developer has already created, laying them out and configuring them in new ways.

However, there is one important challenge you must ensure you can address before enabling embedded self-service: upgrading customizations. If your users can create their own objects - reports, dashboards or new configurations – then, when you upgrade the application, the platform must detect and respect those customizations. Otherwise, users can lose their carefully crafted work, or the hosting application may break in unpredictable ways.

# Summary

Table 3-1 summarizes the main points of difference between the options described above.

Table 3-1: Architectural platforms for embedded analytics

|  |  |  |
| --- | --- | --- |
| Architecture | Benefits | Weaknesses |
| Component libraries | User experience can exactly match the host application. | Every component must be customized in code.  Security and administration must be coded for each component.  Maintenance can be complicated to implement and test. |
| Enterprise reports | Centrally governed.  Highly scalable.  Standardized across the enterprise. | Long development cycle.  Restricted user options or customization.  Limited interactivity. |
| Business Intelligence applications | Rich palette of visualizations.  Good range of connectivity.  High performance from in-memory engines. | User experience can be very different from the host application.  Highly interactive environment can confuse or distract users. |
| Embedded analytics platforms | Highly customizable: even enabling some self-service.  Optimized for embedding.  Integrates with application security and administration. | Higher cost than Component Libraries.  Requires more training than Component Libraries. |

In this chapter we have looked at the common elements of an embedded solution and considered some of the advantages and disadvantages of different approaches. Some of these are so important they require a more detailed consideration, so in the following chapters we will look at Data and Security as elements critical to your success.

4

Data for embedded analytics

One of the first questions we have to ask when building an analytic solution which will be embedded inside an operational application is, “Where will our data come from?” And the answer to that is not always simple. Your data will not necessarily come from the operational application itself. In fact, very often we want to complement or augment the application with insights from other systems.

As we do so, we have to think of three different scenarios for each data source that we wish to use.

* The first is simply where does data come from? Which system has the best data to support our decisions or to support the application that we are enhancing?
* The second question to ask is how will we connect to that data? There are different ways of connecting (even to the same data source), which may be more or less appropriate for the scenario in hand.
* And the third question to ask is how will we manage and govern the connectivity, the security, and the performance of the connection to that data source?

In this chapter, we will look at several different data sources that can be useful for embedded analytics and we will consider those questions What is the nature of the data source and why would we use it? How will we connect to it?

# CSV and other text files

There are few data sources as simple – in theory – as a text file of data. They are found in almost every business in some scenario because they have some great advantages. For one thing, they are widely supported by data analysis tools ranging from the simplest spreadsheet to the most advanced data science workbench. CSV files are equally well supported as an export format. Connecting to a CSV file typically only involves having read permissions to access the folder in which it stored. So for many situations, they are a convenient solution.

This convenience comes at a cost, however.

**Security**. The only security mechanism for CSV files is to protect access to the folder in which they are stored. There is no support for row or column protection. It is important therefore that if you are using CSV files as a source they must be exported with the strictest security and privacy controls in place. Do not save sensitive data to files.

**Performance**. Most data movement tools, such as ETL applications for data warehouses, are optimized in some way for text files, especially for reading or writing data in bulk. These optimizations often take the form of read and write buffers which load data into memory for parsing. Business intelligence tools rarely have these optimizations in place. They will read text files easily, but performance is likely to be poor compared to a native application driver.

**Parsing**. Not all text files are CSV files – comma separated values. Some are delimited by tab characters, some by pipe or vertical bar characters. Highly customized delimiters are rare. And, ironically, ASCII control codes, intended to be a standard, are rarer still. Numerous issues can arise, such as the use of decimal separators (which in some countries are commas) or less common thousands separators, such as the Indian Lakh (1,00,000.) If your analytics environment does not support these characteristics, you will need to ensure that your data export process renders the text files in an easily readable format.

# Operational data sources

An operational data source is simply the data which is stored by an operational application. All modern applications to some extent, create, store and manage data. For example, an application for processing orders generates the data for each order, and perhaps links data for each customer. That will need to be persisted somewhere. Operational data sources focus on integrating that data in such a way that the performance of the application is maximized and the integrity of the application is ensured. In many cases, the ease of use – especially the responsiveness of the application - is enhanced by having a robust data store which can persist large volumes of data for record keeping with good performance.

Many applications use very simple structures for operational data sources. For example, many use an underlying operational database. Developers are familiar with the connectivity to that database and they're quite familiar with the table structure, because typically they're working with the details of the operational application daily. And because you're connecting to your operational data source, consistency is pretty much guaranteed.

Sometimes, you may use a copy of the operational source. For example, you may take advantage of a technology such as replication. The advantage of that for a developer is that the work they do with analytics doesn't affect the operational application itself. So, for example, you could extract all the records for an individual customer, aggregate them do some analysis on them and present that to the operator. You don't have to look up all the details of the original database because you're using a replicated copy which improves performance and preserves the consistency of the original data source.

As we said, operational data structures are optimized for efficiency and for integrity. But the implication of this is that they are not optimized for analytics. In practice, the joins between tables in a highly normalized database can be very complex, tricky to navigate and often don't provide the best performance. In such cases trying to execute analytics over these tables becomes an extra load on the database. Replication can help with that, but you may find it better to use another system such as the data warehouse, which we will discuss later.

However, when it comes to the connectivity to an operational system, this is typically very simple because your operational application is already connected to it! When we connect to a replicated version of the operational data source connectivity will also typically be simple because the replicated version is just another copy of the original system.

The situation does get a little more difficult if we are trying to combine two operational systems: one in which we are performing operations, and one from which we are sourcing data to enhance the analytics.

You may have a purchase management system which handles orders and invoices and various aspects of the order-to-delivery process. The customer data may be in quite a different application: it may be in a CRM system, or a financial system. Bringing those two together involves embedding one within the other, but they're both operational systems and therefore not particularly optimised for analytics. Our connectivity to the second operational system may not be suitable.

In fact, for any operational system, there are typically two ways to connect: through a bulk connector or through an API. A bulk connector is optimised for extracting a large amount of data. An example of this could be connecting with OLEDB or JDBC, to the operational database underlying the application. With such a connector, you can ask for a full table of information, or for many tables at a time, and you can probably issue a query against it and return a result set which may be small or actually very large. That kind of connectivity can be very useful for analytics because typically, we're looking at larger volumes of data to analyze.

Nevertheless, we also need to consider the other way to connect to operational systems: through API's if they are supported. Web developers will be very familiar with this, because most of the connectivity between web applications involves API calls. However, API's are typically not designed for returning large volumes of data: often only single records or the answer to a simple query and the result set may be returned as an XML document rather than as a flat table.

As a result, we have to be careful in choosing the kind of connectivity that we want to use for analytics. It will not always be the case that an OLEDB connector is the best one. Many of them are very simple and performance may not be good. Equally, it will not always be the case that the API is inappropriate - some API's do have calls which enable bulk data extraction. In other words, we have to know our system pretty well, to understand the volume of data we are trying to extract, the characteristics of the source system and the workload we ask it to deliver for us.

# Analytic data sources

There are three types of data sources we might consider as analytic.

## Data Warehouses

The first and most common example would be the data warehouse - an architecture specifically intended to enable data analysis. Its data is denormalized to make it simple to query and to improve the performance of large analytic queries. The architecture is designed to integrate data from multiple data sources so that work does not need to be repeated. Most importantly, I think the data warehouse represents not just a data structure, but a business model. That is to say, a logical representation of how a business operates.

A data warehouse will have dimensions which represent how the business thinks about its customers. It will include a Geography dimension, which represents how the business thinks about its divisions and its offices and it stores our factories. A data warehouse nearly always includes a Time dimension which represents the calendars, working days, holidays and seasonal divisions relevant to the business. It may include a Personnel dimension, which reflects the structure of the company's hierarchy. All of these pre-designed and pre-populated dimensions, make analytic queries very simple for the developer. Even better, the data warehouse should be optimised for analytic performance.

In fact, a data warehouse or it's simpler, more focused cousin, the data mart, is an excellent source of data for embedded analytics. But not every enterprise has a data warehouse and certainly not every small and medium business. Even those enterprises which do have a data warehouse may not want to make it available for operational applications, because in some cases the warehouse is so focused on management, financial or tax reporting: mission critical business analysis, which may be sensitive or governed in such a way that the business does not wish to make the data directly available to other systems.

Sometimes, we can work around that by creating downstream replications of specific subject areas and even the answers to specific queries which we regularly used.

## In memory engines

The data warehouse technology with which we are familiar has been with us since the 1990s.

Around 2005, another technology started to appear, which had many of the advantage of a data warehouse, but offered greater performance, greater flexibility and had advantages for both large and small businesses. This was the development of in-memory engines, which took advantage of the very high performance of memory as compared to disk for performing complex queries. With 64-bit systems (and cheap memory chips) emerging in the mid 2000s, this enabled companies to perform analysis on the desktop, which previously had only been possible on the largest corporate systems.

Building on these desktop capabilities, in-memory technologies are very popular in self-service business intelligence, and in fact, are the key to performance and flexibility for many of the most popular self service applications, such as Microsoft Power BI, Salesforce Tableau, and Qlik. However, self-service applications were in the past very self-contained and focused on their own analytic capabilities. Their connectivity to other systems was entirely focused on extracting data into their own memory space. Developers didn't think very much about how they would share that data with other systems. Having said that, many of the in-memory BI applications today now have embeddable capabilities. So, it is more often than not, possible to take a chart from a BI tool and embed it inside an operational application. There will be benefits for developers in terms of familiarity and performance and simplicity. But there are some restrictions because the systems are often not designed from the start to be embedded. Originally, the embedding ability (and often the API's) are afterthoughts, which have been added later. Nevertheless, for developers familiar with these systems, and where the systems are already deployed in the business or in the enterprise, it's a very attractive proposition to reuse this functionality and embed it inside operational applications. This scenario is often well supported by the vendors and their communities. And it's an option that is well worth considering.

## Data lakes

The data lake architecture, and perhaps in its more modern incarnation, the data lake house is, like the data warehouse, focused on the storage of large volumes of data, which we make available for purposes such as data science, and to some extent analytics.

Unlike the data warehouse, the data lake does not have a built-in business model, a logical semantic architecture, which represents how the business thinks about itself - with rules, well-defined processes for governance or easily-navigated data structures. In fact, the data lake is intended to store data in its most raw format without any processing. This is a great advantage for data scientists, who often want to see data in its native state, because they can use that raw data to do more detailed and more complex analyses that they could if the data had already been cleaned up and built into a data model for reporting.

In the data lake, the data scientist can see all the raw detail - with all its errors and complexities.

# Data Integration pipelines

Another source of data for an embedded system may be the output of a data integration pipeline. That is to say, rather than reading data from a table in an in-memory system or a data warehouse or an operational database, that data will be read from a pipeline running in a data science environment. The pipeline may perform numerous operations of integration, cleansing and preparation on the data from whatever source it comes. This is a very popular scenario for data science. But it is limited in use for embedded analytics, because the process is a little more fragile and a little more difficult to govern than a data warehouse or an in-memory system. This is because the pipeline is more dynamic and more volatile, only delivering data while it is running.

A scenario where this is popular and useful is when we are presenting data which is continuously changing - streaming data. This could include data from machines, such as automated lathes, in manufacturing, or an e-commerce ordering system where hundreds of orders may be placed every minute, or from environmental sensors.

In such cases, the analytic scenario is somewhat different. An application offering streaming analytics must include methodologies for handling interruptions to the stream. Similarly it must be able to display changing values without users having to manual refresh the display. Finally, methodologies for governance and security may be different from the analytic governance that we're used to.

# Writing back to sources

A final point has to be made about the handling of data in embedded analytics. We have been talking so far about data extracted from source systems and presented in the embedded analytic environment purely for the purposes of information and augmentation. This is a read only scenario. There are times when we may want to make changes to the analytic system from within the embedded application - what we call writeback.

Writeback is often used for planning and simulation applications, where we want to insert numbers that represent future estimates of sales or staffing or growth, or whatever the parameters are that we are planning around. It is relatively unusual for systems designed for analysis – such as a data warehouse to enable direct writeback. Such systems are too concerned with data integrity, data history and compliance to allow ad-hoc changes. But often they will allow writeback to special tables, which can be set up as parameter tables specifically for the purpose .

# Summary

We have seen that data can be sourced for embedded analytics from many different kinds of systems: operational analytic and streaming. There is no one best solution for a typical embedded scenario, but it is fair to say that most developers of embedded applications will start with a simple data warehouse or data mart scenario if that already exists within your organization.

The following table summarizes some of the benefits and weaknesses of the approaches we discussed.

Table 4-1: Data for embedded analytics

|  |  |  |
| --- | --- | --- |
| Data Source | Benefits | Weaknesses |
| CSV Files | Widely supported, often human readable. | No security mechanism.  May be difficult to parse.  Performance may be poor. |
| Operational database | Developers are familiar with the connectivity and table structure.  Because you are connecting to operational data, consistency is guaranteed.  If replication is available, you may use a copy of the operational system for analytics. | Operational table structures are optimized for efficient transactions, not for analytics. Joins can be complex.  Analytics is an extra load on the database and can affect performance, although replication can help. |
| Views and aggregate tables | Analytic queries are simplified because the joins were developed in advance.  Filtering is simple.  We can apply role-based access rules simply or build them into the view. (Like the Secured Views of some cloud data warehouses.)  Aggregating data in advance improves performance. | Response time may still be slow if the underlying joins are complex.  Views and aggregate tables must be maintained separately from the operational tables: another burden for administrators.  Aggregate tables must be updated, often using triggers. These updates have to be carefully scheduled or they will affect the performance of the operational system. |
| Data mart or Data warehouse | High performance as the architecture and schema is optimized for analytic queries.  Simple to integrate with BI and analytics tools as the schema is designed for analysis.  Load on the transactional source system is reduced. | Loading the data warehouse using ETL is a specialized practice.  The load and refresh cycle of the warehouse (often scheduled to run overnight) may be too slow for analytics embedded in an operational application.  When schema changes are needed, this can be a complex process requiring a lengthy process of specificying, implementing and testing. |
| In-memory engine | Very high performance.  Should be closely integrated with the embedded analytics environment. | Support for writing back to the database is often limited.  Data updates may be occasional and often require full, rather than incremental, updates. |
| Data lake | If using a data “lakehouse” architecture, you will find many of the benefits of a data warehouse.  Data is generally stored in its native format which can be good for predictive analytics.  Data may be continuously updated. | If you are not using a lakehouse model, you may find performance for business intelligence poor because so much transformation is required to conform and shape data.  Data lakes are generally shared with data scientists, data engineers and analysts. Security can be a concern. |

In general, I recommend starting with a data mart or data warehouse connection if this already exists in your organization’s architecture. You’ll find it gives the best balance of accessibility, performance and governance for your needs. A data lake or lakehouse, if well managed, offers some similar advantages. However, if you don’t already have a formal data warehouse or lakehouse in place, it is a significant effort to build one and likely beyond the scope of your embedded analytics project on its own.

In this case, you may find it practical to start with views and aggregate tables, defined for your purposes and hosted within the operational database. The administrator can provision access and ensure security and good governance, performance should be acceptable and data access should be quite possible with a suitable driver.

Having reviewed the data available, in the next chapter, we’ll consider the different types of embeddable objects and how we can integrate them visually with applications for the most effective support.

5

Embedding analytic objects

Now that we’ve discussed the potential data sources that can power embedded analytics solutions, we can consider the actual *objects* that we can embed. In this chapter, we’ll talk about the various types of analytic objects, how users and consumers interact with them and what considerations – technical and otherwise – should be evaluated when planning, developing and implementing embedded solutions.

We’ll also explore the ways in which embedded objects can be visually integrated into third-party applications to provide a seamless experience for end-users. We will also cover specific technical processes that extend functionality and allow embedded solutions to integrate with other Enterprise Resource Planning (ERP), Customer Relationship Management (CRM) and communication platforms.

Note that in this chapter, we’re being deliberately broad by using the term ‘object’ to refer to any analytical element, component or content embedded within a host application, whether it’s served from within the host application itself, or from an external source or application.

# What can we embed?

It’s worth outlining some ways to categorize types of analytics objects that can be embedded in any solution. First, we should remind ourselves that embedded analytics isn’t just about integrating analytics objects, but also analytical *capabilities*. These capabilities – like Natural Language Generation (NLG), for example – help end-users make better decisions in the same way as more typical embedded objects like charts and graphs, but often rely on very different technical approaches. Second, whilst categorizing objects in this way is useful, bear in mind that truly effective embedded analytics solutions will often combine these capabilities to provide better accessible and increased adoption for less technical users.

Let’s look at some examples.

## Key Performance Indicators (KPIs)

From an analytics standpoint, the ubiquitous KPI is one of the most common terms you’ll hear when dealing with embedded analytics. It simple terms, it represents a measurable value that evaluates progress towards a previously agreed objective, like a sales target, response time or quarterly revenue. Although KPIs can be considered as a type of data visualization (static or otherwise), I’d encourage you to consider them as a distinct type of object for a couple of reasons:

1. They’re almost always *highly* *aggregated* measures or give a very simple visual indication of what’s happening to an aggregated measure. Is this figure positive or negative? Are sales up or down? This is particularly important in the context of embedded analytics, where quickly giving end users some level of directional insight means they can ‘stay in the flow’ without spending a long time interacting or interpreting complex charts.
2. The nature of KPIs means that they are a powerful way of presenting a lot of high-level insight in a relatively small area. Again, this is useful for embedded use-cases, where room on the page or within and application may be given over to other content, not just analytics.

Text

Description automatically generated

Figure 5-: A section of a dashboard illustrating various KPIs

Figure 5-1 illustrates a very typical use of KPIs to summarize the performance of three metrics. Note that KPIs are often augmented by other indicative objects to add some context – in this case, our SALES YTD KPI shows us an aggregated sales value, along with the change in that value (expressed as a value and percentage chance) compared the same period a year ago. There’s also a color-coded indicator to help us understand if the change is an increase or a decrease. Clearly, this approach builds on the basic idea of a single-metric KPI and starts to use some more complex data visualization techniques.

## Data visualizations

Humans are pattern-recognition machines. Interpreting data visually gives us less cognitive work to do, and our brains are better suited to it. We also tend to process visual information in very predictable and consistent ways, so it makes sense that data visualizations are the mainstay of any embedded analytics solution.

There are thousands of resources, books and websites out there devoted to the topic of data visualization (as of the time of writing this, Amazon brings back over 7,000 results for the term) which go into exhaustive detail about the history, creation and interpretation of charts and graphs. However, for our purposes, let’s summarize some key types of data visualization objects you’ll see in embedded analytics solutions by thinking about what they’re *for*, as well as what they look like.

Chart, bar chart

Description automatically generated

Figure 5-: An example of a bar chart

Column and bar charts, shown in Figure 5-2, are most often used to compare the size (or magnitude) of things, like total sales across regions.

Chart, histogram

Description automatically generated

Figure 5-: An example of a histogram

Figure 5-3 shows a histogram. They’re often used to show how often things happen and are the standard way to show statistical distribution, like the breakdown of a population by age category.

Chart, line chart

Description automatically generated

Figure 5-: An example of a line chart

Line charts and their derivatives (area charts, step charts, etc.) are used to show how something changes over time, like daily call volumes in a call center. Line charts are often the basis for trend charts.

A picture containing diagram

Description automatically generated

Figure 5-: An example of a Gantt chart

Gantt charts are of particular relevance to embedded solutions where the user is already working in an ERP or CRM application. They are useful for visualizing the duration and relationship of things over time, like tasks in a project.

Chart, bar chart

Description automatically generated with medium confidence

Figure 5-: An example of a stacked bar chart

Stacked column and bar charts are often used to show the relationship of part-to-whole data, like the breakdown of budget spending by category. The most common (and controversial) type of part-to-whole data visualization most people are used to is the pie chart.

Map

Description automatically generated

Figure 5-: An example of a spatial chart

Spatial charts are used to visual data in relation to specific locations or geographic areas, like plotting locations on a map expecting rain.

Chart, scatter chart, bubble chart

Description automatically generated

Figure 5-: An example of a scatter plot

Scatterplots are used to show the relationship between two variables, like the correlation between age and shoe size.

Naturally, many other chart and graph derivatives and combinations exist, along with more esoteric chart types like Radar charts, Isotypes, Sankey diagrams, Chord charts – all of which usually have specific use-cases.

Theses chart types can be very powerful at bringing insight to end users, but I’d advise anyone stepping into the world of data visualization to remember that a chart or graph is only as effective as a viewer’s ability to understand it – or indeed *interact* with it, something we’ll go into more detail about later.

## Tabular data

So, what do we mean by the term ‘tabular’? As an all-encompassing definition, we can start with ‘data presented in a table’, which for most of us would mean a spreadsheet, like an Excel document or a Google Sheet. As mentioned in previous chapters, spreadsheets remain an executive tool of choice, and they’re often the most common and fundamental way that users will encounter analytical content within applications.

Let’s look at a few examples.

Table

Description automatically generated

Figure 5-: Guardian graphic | Source: CovidLive.com.au, Australian Bureau of Statistics, Guardian Australia

What we see in Figure 5-9 is a great example of a summarized, tabular analytics object embedded in a website, showing summarized data about Covid vaccination statuses in various Australian jurisdictions. As with our KPI example, this table also includes data visualization techniques (color-coded heatmapping computed at the column level) to help readers identify the highest and lowest values.

Table

Description automatically generated

Figure 5-: Section of a page in a CRM system

You’re also highly likely to encounter tabular objects in CRM and ERP systems, like what is shown in Figure 5-10, an example of a Salesforce page which summarizes yearly sales performance figures. Note that there’s a high level of precision for the aggregated figures in the bottom section. Whether or not this is useful to the viewer is a decision you need to make when using any kind of aggregated values shown in a tabular format.

## Dynamic text and NLG content

As the capabilities of data and analytics applications continues to increase, text-based content is often used to augment embedded analytics and present insights in-context for users to read without needing to interpreting data visualizations.

Text

Description automatically generated

Figure 5-: An example of data-driven text

In Figure 5-11, static (pre-determined) text is augmented with four dynamic (data-driven) fields that update based on a data source. The advantage of this approach is that it requires low cognitive effort to interpret the data and the text can be tightly integrated with other non-analytical content in an embedded solution. Depending on the way this is visually structured, the user may not even be aware they’re interpreting analytical data at all – potentially an impressive achievement.

Going beyond dynamic text, Natural Language Generation (NLG) technology uses machine learning techniques and computation linguistics to produce written text from data sources without the need to pre-determine structure or syntax. As of the time of writing, this fast-evolving technology is often presented as a complimentary technology alongside many of the leading data visualization vendors, both to add value to traditional data visualization dashboards for end users, and as an aide to statistical analysis – allowing more technical users to quickly identify significant trends and outliers.

Text

Description automatically generated

Figure 5-: An example of NLG

In this example, all the text has been generated by an NLG system, including the grammatical structure and presentation of language as well as the data-driven metrics. If well configured and carefully deployed, you can imagine how powerful this kind of approach can be – and home much potential time it can save – within embedded analytics solutions.

# Adding interactivity

As we have mentioned already, embedded analytics isn’t just about integrating analytics objects, but also analytical *capabilities*. NLG is a great example of embedding an analytical capability, but it’s not something immediately obvious to an end-user. One of the most effective ways to provide additional capability is to embed objects that can be interacted with.

## Interaction examples

When we talk about interacting with analytics objects, we simply mean providing some element of human input: clicking a button, dragging a scroll bar, entering a value into a search box. Here’s some examples:

* **Hovering** over an object to reveal some additional or contextual detail about a data point on a chart. Tooltips are the most common examples of this, but this technique can also be used to trigger other actions, like highlighting related objects or areas on other charts.
* **Selecting** data points on a chart in order to filter related objects on other charts. Selections on data visualizations can often be made in the same way as most user / application interfaces allow – dragging, multiple-selections, etc.
* **Drilling** into (or ‘down’ or ‘through’) a single element in a data visualization in order to expand the constituent elements. Whilst driven by a selection, this is an extremely common technique in embedded analytics solutions to allow users to move from KPI or highly aggregated objects into more granular or row-level views for more detailed analysis.
* **Navigation** using traditional methods like tabs, menu links, icons, etc. is another common interaction technique used to allow users to move through a sequence of data visualizations, perhaps embedded in a single page.

Of course, the expectation is that whatever you’re interacting with updates quickly to show the result of your input, and – crucially – adds some additional capability or value to the embedded object(s). However, it’s entirely possible to use lots of complex interaction techniques and end up with complicated, unintuitive and meaningless analytics experience. Let’s look at an imaginary example that makes best use of some of these techniques:

Graphical user interface, application, table, Word

Description automatically generated

Figure 5-: an illustration of an interactive dashboard embedded in a CRM system

In Figure 5-13, the embedded object (a dashboard) is displayed as a menu option, tightly integrated within a CRM system. The user has access to other areas of the CRM with a single click, and the analytics content is this section has been created with a visual style that matches the CRM system.

Within the dashboard itself, we’re making use of several techniques we’ve already touched on. The user can **hover** over a data point in the bar chart to see a tooltip containing some more detail information; they can **click on a bar to filter** the details in the grid, and they can also **apply custom settings** that will affect the way the KPIs are calculated. The fact that this is a Summary dashboard is clearly indicated with navigation buttons, and the user can **navigate** to additional dashboards as needed.

## Interaction as a value-add

It’s also useful to think about interactivity from a *product experience* perspective, putting yourself in the position of an end user and thinking critically about the actual use-case. Will a representative user *want* to interact with visualizations? Will they *need* to interact in order to get actionable insights, or is a simpler method more appropriate?

Many embedded analytics solutions use interactivity as a unique selling point (USP) to support higher-cost access for users and consumers. You can easily imagine a free ‘bronze’ plan that provides access to a limited number of non-interactive objects, versus a paid-for ‘gold’ plan that provides access to a broader range of fully interactive objects.

Designing a system that offers a great experience to its users applies to embedded analytics solutions, just like every other user interface.

## Technical considerations

For interactive analytics to provide value, interaction response times must be fast.

In a perfect world, to create the illusion that a user is directly manipulating something on a screen, a response time of less than 0.2 seconds (200ms) is required. This might seem ambitious but bear in mind that many common interactions with pre-loaded objects like buttons, drop-down lists and link behaviors on a typical website are usually within this range.

However, in the context of embedded analytics we might need to lower our expectations. It’s true that many analytics tools have ‘instant’ responses to user interactions for native features like tooltips, drop-down filters and selections. But if the interaction generates a query back to the underlying data source (or otherwise requires new data to be loaded and rendered), there’s going to be a delay and it’s likely to be longer that 200ms. So, when planning an embedded analytics solution, what can we do?

* Consider the impact that infrastructure and solution design might have on response times. Optimized, denormalized data extracted from your data mart or data warehouse (which lives close to your analytics solution) *may* perform faster than a live connection to table hosted in a SaaS database. Similarly, optimizing and right-sizing hardware for analytical workloads can yield great results.
* Understand the caching model that analytics vendors use, and how it may impact your solution. Most database, data warehouse and data lake vendors have complex and sophisticated query and result caching, which can often be used in conjunction with data visualization vendor caching to only generate new queries or page loads when it’s necessary, thus reducing response time to an absolute minimum.
* Define upper and lower bounds for interaction response and monitor them. Knowing what “good” looks like leads to efficient optimization and management – if a 5-second response time is likely to be sufficient for your use-case, then efforts can be targeted on only the worst-performing specific objects, pages and visualizations.
* If possible, test the impact that concurrent users have on your solution. Concurrency can be complex to define but having some understanding of how response times are affected by many users who are simultaneously interacting with the same object(s) can very useful – especially when scaling out embedded analytics solutions to support large numbers of users.

In general, having a good grasp on the technical architecture of any planned solution will likely surface many of these considerations during evaluation.

# Embedding objects with iframes

So how do we embed analytics objects in our target application or webpage? Let’s explore one of the most common methods used by almost every analytics platform that support embedding of visualizations: the iframe.

Note that we’re talking about *true* embedding here – i.e., embedding content from a third-party source (like a data visualization tool) in a target application or website (like a CRM or ERP platform) be it accessed via a browser or mobile application. Clearly, some analytics objects may appear to be embedded in a target application, when in fact they’re actually integrated within the application itself.

Embedded analytics solutions – where the content or object being embedded is being served from a different system to that of the page it’s embedded into – will almost always use an iframe to do the ‘embedding’.

We’ve made mention of these already in previous chapters - an iframe is a standard HTML element that can be used to load another webpage within the parent page. Most popular data visualization applications have features that allow content to be embedded within iframes, either directly using URLs, or by invoking APIs to request content. This methodology can also be used on mobile apps using [WebView](https://developer.android.com/reference/android/webkit/WebView) components.

Here's a very simple example of an HTML page with an embedded iframe and the resultant view a user would see:

<h1>Here's a web page in an ERP system</h1>

<p>This content is part of the ERP system</p>

<iframe src="https://en.wikipedia.org/wiki/Main\_Page" width="100%" height="300"></iframe>

<p>This content is also part of the ERP system</p>

Graphical user interface, application

Description automatically generated

Figure 5-: an example of an iframe embedded in a webpage

In Figure 5-14, we’ve applied a little styling to our HTML using CSS for clarity but note that everything within the red box is content we’ve loaded into the iframe – in this case, the Wikipedia homepage. At a high level, this is exactly how embedded analytics objects are embedded in target systems.

## Using iframes to our advantage

One of the key things to consider is that when an analytics object is embedded, the end-user experience is likely to be much more customizable, which is a great advantage for usability. Here are the benefits of doing this:

* Content to help users understand how to use the embedded content – FAQs, user guides and tips – can be displayed alongside or in context with the embedded objects.
* Narrative content explaining interesting conclusions or insights derived from the embedded objects can be tightly integrated in an editorial fashion – many major news outlets use this approach to support data-driven editorial content.
* Branding can be applied *around* the embedded objects to further improve the look and feel of the solutions. If you’re embedding a data visualization, it’s entirely possible that the title and description of the content can live outside the iframe, on the parent page – and therefore formatted accordingly.

These are only basic possibilities – we’ll talk more about the potential look and feel and user experience of embedded solutions later in this chapter.

## Cross-domain limitations

Most modern web browsers have security mechanisms that restrict how resources from one origin – the web page you’re browsing – can interact with resources from another origin – a visualization from your reporting platform displayed in an iframe on the same page, for example. The principle behind this mechanism is sound: you don’t want a random website running some suspicious JS code against your webmail in an iframe embedded on the same page (loaded without your consent). This security mechanism is specifically designed to prevent the ‘cross origin’ content from loading in the first place.

Why is this a problem for embedded solutions? In many cases, the reporting platform that serves up analytics objects may not be from the same *origin* as the website or app you want to embed them in.

The specifics about exactly what constitutes a different origin are beyond the scope of this book, but for our purposes we might consider a simple example where your website at **http://my.business.com/** has analytics objects from **https://mi.reporting.com/** embedded in it. In many browsers, this would constitute a different origin because the embedded content is not only from a different *host*, but it’s also being served from a different *protocol* (HTTP instead of HTTPS).

Fortunately, whilst outside the scope of this book, there are ways we can resolve cross-origin challenges, such as ensuring that both domains are explicitly trusted, or using methods like [CORS](https://en.wikipedia.org/wiki/Cross-origin_resource_sharing) to specify which additional origins should be permitted to load resources.

# Current embedded analytics trends

Embedded analytics solutions are at the heart of many data monetization products and are becoming increasingly sophisticated as organizations take advantage of SaaS products and other cloud services. This rapidly innovating sector has generated some interesting trends and consolidation patterns:

## Using embedded analytics to share data

Many embedded analytics products are designed around the concept of self-serve, which we’ve touched upon in previous chapters. However, it’s often inevitable that such products need to allow users to select, export and download data for other uses. In these cases, embedded objects are often used as ‘selection mechanisms’ to identify and filter a selection of data in order to be exported.

As an example, consider a user drawing a selection over a map in order to build a selection of ZIP codes. In this case, the user is limited in their selection by the interactivity features in the visualization, but more sophisticated methods are available. With appropriate user authentication and authorization, it’s entirely possible to allow users to write Data Manipulation Language (DML) statements in SQL to select whatever they want from the underlying data source *alongside* the embedded options that uses the same data – potentially a very powerful value-add proposition for an embedded solution.

## Transformative best-practice visualization

Whilst adding more sophisticated features to an embedded analytics solution undeniable adds value, observing best-practices for the data visualization itself is often overlooked as a way to differentiate the product. What do we mean by this? Here’s some examples:

* Using the most appropriate type of visualization for the data
* Avoiding misrepresentation of data, like truncating axis to exaggerate comparative dimensions
* Careful application of color schemes, spacing and typography

Observing these principles *will* make your product easier to use and more engaging for users.

## The look and feel of embedded experiences

This is where the capabilities of the tool you’re using to build analytics objects really comes into play. Many data visualization tools are highly customizable and allow the user to build interactive experiences that are more akin to ‘dynamic infographics’ – very tightly integrated into the target system, augmented with narrative, imagery and behaving more like an application than a traditional dashboard.

Tableau have a public website that showcases some of the best example of dynamic and interactive visualizations created using the tool. It’s available online at <https://public.tableau.com/app/discover/viz-of-the-day>.

# Putting it all together

We’ve talked about the types of analytics objects you can embed, and how capabilities and value can be added by considering some of the human and technical aspects of embedded analytic solutions. Let’s think about what we’ve learned in relation to some real-world use-cases:

## Embedding workflow

A core principle of embedded analytics is about keeping users ‘in the flow’ when making decisions informed by data. The fact that the analytics object is *in* the tool you’re already using means you don’t have to *leave* the tool, which certainly helps – but as we’ve learned the nature of the visualization, how fast it reacts and how effective it is for the user are all significant factors.

This means it’s important to think about the context of that workflow. Where has the user been before they get to see the embedded content? What will they do once they’ve interacted with it? How can the embedded objects drive upstream and downstream processes that might be integral to the “flow” we keep talking about?

## A typical reporting automation workflow

This is an illustration of a basic reporting workflow. A finance team needs to gather data to provide a monthly report and requires the involvement of the IT / database team to do so. Once they’ve got the data they need, they create (or most likely, update) a pre-existing document, and provide that to the management team to use a decision-making tool.

What do we get with this process? Delays whilst different teams are involved. Repetition of effort. Potential for error. Unmanaged proliferation of content, and potential security problems too. In short, a suboptimal pattern that’s probably happening in thousands of organizations right now.

Diagram

Description automatically generated

Figure 5-: An example of resource-intensive and costly analytics workflow

Figure 5-15 illustrates this convoluted, multi-dependent process as it might apply in a typical organization. This is often the first real-world use case that gets addressed with embedded analytics. When we incorporate an embedded analytics workflow, as shown in figure 5-16, the analytics content is integrated into an organization-wide ‘portal’, where analytics objects are driven by data from the organization’s finance database and only accessible to the management team. A given manager need only to log in, review and interact with the analytics object (a dashboard, for example) and make an informed decision.

The contrast is obvious – there is a single report, which doesn’t need to be updated by anybody. The data is accurate, access to it is tightly controlled, and the reporting content has not taken any time to be created – in fact, the finance and IT / database teams haven’t been involved at all.

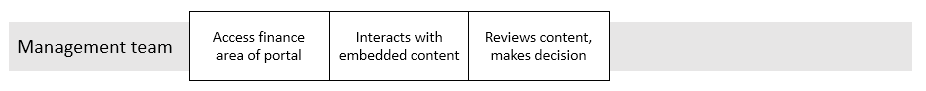


Figure 5- : Using embedded analytics to accelerate time to insight

The potential time and cost savings realized by using embedded analytics in this way are non-trivial.

A core principle of embedded analytics is about keeping users ‘in the flow’ when making decisions informed by data. The fact that the analytics object is *in* the tool you’re already using means you don’t have to *leave* the tool, which certainly helps – but as we’ve learned the nature of the visualization, how fast it reacts and how effective it is for the user are all significant factors.

This means it’s important to think about the context of that workflow. Where has the user been before they get to see the embedded content? What will they do once they’ve interacted with it? How can the embedded objects drive upstream and downstream processes that might be integral to the “flow” we keep talking about?

## Using embedded analytics to power prescriptive analytics

Prescriptive analytics is the term used to describe an analytics model where the solution specifically describes the actions needed in order to achieve a particular outcome. It’s simple to describe, but often involves a lot of complexity – not least because most prescriptive analytics techniques involve presenting non-technical users with the results of machine learning processes, which is not an easy feat to accomplish.

When embedded analytics solutions are implement successfully, it becomes possible to integrate prescriptive analytics processes to provide a seamless experience for end users.

Here’s an example:

1. A sales manager accesses an embedded chart in their CRM system indicating that their sales figures are surprisingly low for the current quarter. They contemplate addressing the problem with the sales team.
2. A machine learning model which has been trained on historical sales figures over the last five years determines that in order to meet the current targets, the best option would be to focus efforts on reducing the price of three *specific* product categories that have historically generated strong sales with appropriate discounts applied.
3. The specific discount percentages are determined for each product categories, and the recommendation is shown alongside the embedded chart in the CRM system as the optimal way to address the poor sales performance.

This is truly closing the loop. Rather than addressing the issue with the team using intuition and guesswork, the sales manager can issue a single, statistically proven directive: *discount these three product lines and we’ll be back in business.*

It’s worth noting that prescriptive analytics is rarely deployed effectively due to the complexity of processes involved and the requirement for advanced analytics and data science / machine learning (DSML), which often require highly skilled data scientists or analytics engineers to enact. As ever, software is increasingly democratizing this discipline.

We’re going to spend more time talking about DSML, automation and augmentation around embedded analytics solutions in a later chapter but suffice to say that DSML techniques are often at the center of prescriptive analytics features we’ve discussed.

## Operationalization (or “write-back”) of data

In many cases, the workflows that we’ve described so far may include steps where data needs to be written back into the data source or other operational system to drive further processes. Here are three real-world examples:

* A user tracking dashboard shows that a given user has opened a marketing email. The user record within the CRM system therefore needs to be updated to show that the user has responded to the marketing campaign, which removes them from the recipient list for a period.
* A comment about an outlying data point in a chart is added by an executive, which needs to be displayed as a tooltip against that data point for anyone else viewing the chart in future.
* A user interacts with a dashboard showing errors in a production line. When the user clicks a specific error, a ticket will need to be created in an error-tracking application, containing details about the time and nature of the error.

At a high-level, all these examples work in a similar way. Data derived from the embedded *analytics solution* (either by the actions of the user or derived purely from the data itself) is written back to the *target system* (either a data store within a CRM/ERP system or a dedicated database). In both cases, this delivers the concept of ‘closing the loop’ with an analytics workflow. You use analytics to derive insight, and update the data accordingly, in a single workflow.

These examples all require some level of integration between the analytics solution and the target system(s), but the key point is that operationalization of data can often be driven from within the embedded analytics solution itself, ensuring that the user can *execute* on actional insight without leaving the solution

Operationalization of data as part of analytics workflow is a highly innovative and heavily invested area for both analytics and data warehousing vendors. For those familiar with OLAP vs. OLTP, the technical distinction between transactional use-cases and analytical use-cases has historically necessitated different technical approaches, and therefore, different products.

OLTP use-cases usually indicate small volumes of data *written* into a database very frequently, and OLAP use-cases usually indicate large volumes of data *read* from a database less frequently

With the advent of scalable SaaS offerings and advances in data storage, caching and query performance, this distinction is becoming less prevalent. As of the time of writing, some Data Platform vendors (like Snowflake, for example) are working on unified services to provide databases that can be address for OLAP and OLTP workflows simultaneously – a single store of data that your ERP system writes to *and* the source of truth your analytics application read from.

## Business-case integrations

Integrations between messaging applications (like Slack or Microsoft Teams) and data visualization tools are becoming more common. The objective is to ensure that users can access actionable insights even if they’re *not* currently using the embedded analytics solution. Here’s an example:

1. Vaughn is walking to work, where he’s a supply chain manager at a large retail organization. He receives a notification on his messaging app, telling him that inventory levels for a key product line are below a certain warning threshold, a level that Vaughn himself has configured by setting an alert on an embedded analytics object.
2. Clicking the link in the notification, he’s directed to the relevant embedded data visualization which shows him that an expected supplier shipment has been delayed, with contact details for the relevant account manager.
3. He clicks the link on his phone, calls the account manager and confirms that although the shipment is late, it will arrive in time to support the expected fulfilment volume for the month – which is also shown in the ERP alongside our embedded dashboard. It takes Vaughn five minutes to do this, and all before he gets to his office. He decides to stop to buy a coffee to celebrate.

This example has four integration components:

* A methodology for configuring data-driven alerts within the data visualization platform.
* A messaging app integration that is targeted by the data visualization platform (along with email and other notification channels).
* A carefully designed dashboard, embedded within the supply chain ERP system.
* Appropriate authentication and authorization for Vaughn to interact with these components on his mobile phone.

## Management and governance integrations

Vaughn represents a *consumer* use-case, but there are many other powerful integration patterns possible with many modern data visualization platforms. Many data visualization and SaaS database tools have RESTful or GraphQL APIs that can be interacted with to perform many key tasks related to management, governance and reporting.

**Lineage and impact analysis** can be facilitated by querying the underlying components in a data visualization platform (KPIs, visualizations, data sources, users, events, etc.) to determine the relationship between components. By understanding that a given database holds a table which provides data for a KPI, it’s possible to determine the impact of making a change to that component will have. Will it affect one KPI, or thousands? This information is very useful for teams managing the development of an embedded analytics solution.

Diagram, timeline

Description automatically generated

Figure 5-: An example of how lineage is depicted in Tableau

In Figure 5-17, we can see that 29 fields in a data source are derived from 3 tables in a single database. If I were a database administrator about to make a change to one of the three tables, it’s certainly useful to know that I’m potentially going to affect 29 fields used in a Tableau data source.

Here are some other real-world examples where integration with a data visualization platform (usually facilitated by APIs) can solve common governance and management challenges:

* **Performance auditing.** We’ve made the case that performance is important, and not just in terms of response time to interaction, but also overall loading times for embedded objects. Most data visualization platforms include native content or extensible methods to understand basic performance metrics. How fast are my embedded objects loading? Which objects are interacted with the most? What data sources are most popular? All this information is crucial in order to understand user behavior and to support performance tuning initiatives.
* **Programmatic content management** can be very useful to deploy, migrate and update content in an embedded analytics solution. For example, if your solution contains hundreds of embedded objects and you identify that they all contain a calculation that needs to be corrected, having a method to programmatically update the calculation (and potentially re-deploy / update / publish the corrected objects) will save a significant amount of time and effort compared to the process of downloading and correcting each object individually. APIs can be used to address content in this way in many data visualization tools, allowing content, data sources, users, permissions and group membership all to be managed programmatically. The scope of what you can and can’t change is determined by the features provided by the API, so if you’re evaluating specific tools, it’s always worth researching any related API references to understand what – potentially – could be automated.
* Of particular relevance to external-facing data monetization products, being able to programmatically control **user provisioning** can be very important. If you consider a new customer purchasing access to an embedded analytics product, several things may need to happen:
  + Payment details may need to be captured and processed
  + The user details will need to be captured, and associated with a level of access to the product depending on options selected at the point of purchase
  + The user will need to be set up in any identity provider or authentication system, as well as within the target systems and the data visualization platform
  + Access permissions will need to be applied to control the content the user needs access to. This may extend to objects within the data visualization platform as well as databases, the target system as well as non-analytics content hosted on the target system (files, etc.)
  + Introductory content (emails, FAQs, login links, etc.) will need to be provided to the user.

Your solution may not need all these processes to happen, and indeed it may not be possible to programmatically address some of these processes, depending on the nature of the tools used and the features of any available APIs. However, if you’re planning the deployment of an embedded analytics solution, it’s useful to determine what, if any of these steps can be automated – it will provide long-term benefits as more users are added.

# Conclusion

We’ve now spent some time exploring the components of an embedded analytics solution and talking about the considerations and possibilities these solutions can offer. Hopefully, you now have a better understanding of how to approach the planning, development and implementation of your solution.

In the next chapter, we’ll go into more detail about the administration of embedded analytics – setting up users, understanding licensing models, resourcing management and ownership.

6

Administration of Embedded Analytics

This chapter will explore the challenges of administering analytic solutions embedded in operational applications. In general, there is one golden rule which governs this process …

As far as possible, align all the administration of your embedded solution with the administration of the host application.

This ensures minimal duplication of work and better system coordination. We'll also delve into specific topics such as cloud vs on-premises deployment, security configuration and DevOps integration for a comprehensive understanding of how best to handle these integrated systems.

# Deploying embedded analytics

When it comes to deciding whether to run embedded analytics on-premises or in the cloud the issues involved can seem quite complex. Many small, medium and large businesses are moving to cloud storage as it’s generally seen as the best solution for ease of deployment and cost-effectiveness —but there are some edge cases where it’s not ideal.

## On-premises deployment

On-premises means your company’s server is hosted within your organization’s infrastructure and, in many cases, the server is physically onsite. The server is controlled, administered, maintained, procured, etc. by your company and its in-house IT team, or an IT partner. Data and other information are shared between computers through your local network.

Having an on-premises approach can give you the advantage of having control over your architecture, procurement and scalability. However, there are some drawbacks to this option; it takes longer for hardware to be procured and delivered as well as being more expensive and difficult when upgrading the equipment. Additionally, scaling is not as straightforward compared with cloud hosting where availability is not limited by pre-determined sizes.

## Cloud deployment

Cloud deployment involves having an external service provider, like Microsoft, Google or Amazon manage your data and applications. This includes them taking care of any hardware, software and other infrastructure they need in their datacenters. Access to these services is often from a browser interface, meaning you can control your account and services from almost anywhere on any device.

Hosting in the cloud makes scaling easier because it allows for greater flexibility and scalability than traditional on-premises hosting. Cloud computing offers an elastic infrastructure that can quickly scale up or down depending on demand, allowing organizations to respond more rapidly to changes in their business environment.

With a cloud provider there is no need for costly hardware investments as resources are available when they're needed and don't require upfront capital investment or long lead times associated with physical servers. Instead, all you need is to request a virtual machine or Docker containers. Typically within minutes the cloud provider works to match your requirements, allocating additional server resources such as CPU cores, RAM memory and disk space to accommodate the requested VM/container size. The process may even involve configuring networking rules for it to be accessible over an external IP address or private network segmentation if needed. After these steps have been completed successfully, the newly created instance is ready for use by customers. This simplicity is very attractive!

Although cloud computing offers low costs and simplicity, it can become expensive if not managed carefully. Subscription prices may seem reasonable initially, but increased usage can quickly drive up monthly bills. Additionally, self-service queries created by users are often unoptimized which could result in a great deal of unnecessary data being read and returned at the expense of compute costs.

Diagram

Description automatically generated

Figure 6.1 A typical load-balanced deployment

In Figure 6.1 you can see a typical application deployment, including Single Sign On (SSO) security which we will discuss later in this chapter.

It’s also worth noting the use of a load balancer – such as a load balancing router – to optimize traffic with the internet. A load balancing router is a network device used to manage traffic across multiple networks or servers by distributing traffic among multiple paths to improve performance, reliability, and scalability. Load balancing routers can also be used for load balancing between multiple internet connections or for load balancing between multiple applications running on the same server. Load balancing can be beneficial for embedded analytics, as the load can vary significantly between different clients depending on the user's activity. This could range from simply viewing an embedded report to working with a complex dashboard and visuals or creating self-service objects.

Another approach to deployment which may be easier to manage is to use Containers, discussed later in this chapter.

# IT operations and DevOps for Embedded Analytics

Today most modernized software teams use a DevOps methodology—combining techniques and tools from software development and information technology operations. The aim is to enable more frequent, often continuous, deployments of new features.

DevOps

DevOps is a set of software development practices that combines software development (Dev) and IT operations (Ops). It is designed to enable organizations to deliver applications and services quickly and reliably. DevOps focuses on automation, monitoring, and collaboration to reduce development lifecycles and error-prone manual processes.

There are some disadvantages too. Setting up a DevOps environment can be quite expensive, especially if it requires new investments in software and training. The latter is essential as DevOps requires quite detailed technical knowledge technical knowledge across a wide range of topics and technologies such as source control, automation, observability, testing, infrastructure as code, scripting and virtualization. With all this, as you can imagine, it can also be difficult to adopt a DevOps standard.

However, for Embedded Analytics, the ability to deliver important new features quickly and with more consistent quality - thanks to automated integration and functional testing – is compelling for many teams.

As DevOps becomes increasingly mainstream, it is naturally important that your embedded analytics solution should also support these methodologies. If not, then co-ordinating releases of host applications and embedded analytics may become more complex than necessary and may slow adoption of your solution. In fact, it is common for embedded analytics solutions to get “left behind” by host applications which are updated more frequently and deployed more easily. On the other hand, in early development, you may wish to iterate numerous deployments of your embedded analytics – adding new charts, for example, without disturbing the deployment of the host application.

Here are some best practices to consider when working with Embedded Analytics and DevOps:

**The embedded analytics solution should integrate with your existing security model.**

In general, your Embedded Analytics security will use the same techniques your DevOps team is currently using for the host application. This way, you won’t have to recreate or replicate security information in two different places. Security must be integrated into the entire DevOps process, from development to deployment, ensuring that development and testing environments are properly secured, that code is checked for vulnerabilities, that access to production environments is tightly controlled, and that systems are monitored and patched regularly. Additionally, security teams need to be involved in the DevOps process to ensure that the right security measures are put in place and that security goals are met throughout the entire lifecycle of the application.

**The DevOps process must support a wide range of data scenarios**

A DevOps-friendly embedded analytics solution will not force you to use a proprietary data store, replicate data, or change your current data schemas. Applications incorporating analytics should be able to use data in your existing systems and deliver strong performance across disparate sources, while development is supported by a DevOps methodology. In fact, DevOps with its extensive automation and ability to spin-up multiple environments from script, is well-suited to developing and testing varied deployments.

No matter what the environment, with pre-packaged analytics, you can ensure that queries are efficient and well-defined. However, for self-service applications that enable end users to do their own data discovery, the queries are not defined by the developers. DevOps should be aware of this, so they can test with inefficient and costly queries of the kind that often arises in self-service.

**DevOps should enable you to test against cloud, containers, and hybrid deployment environments**

Embedded Analytics should not make life harder, or close off options, for the DevOps team, now or in the future. So the analytics environment needs to work on multiple platforms and hybrid environments. The DevOps team will look for support for container technology like Docker and orchestration technology like Kubernetes for automating deployment, scaling, and management of containerized applications. These ensure that the transition from development, testing and staging into production on either physical or virtual machines runs smoothly. See the sidebar, Containers and the alternatives.

**DevOps should be able to deploy with their current techniques**

DevOps teams will not want to work with an Embedded Analytics platform that forces them to change their current deployment methodology. So the platform must be flexible, not only in architecture but in deployment, licensing and support for high-availability. For example, if the host application supports load-balancing and failover (perhaps including redundant servers and shared storage devices) then the Embedded Analytics platform should do so too.

**Embedded Analytics should support horizontal and vertical scaling.**

Horizontal scaling is the process of adding more nodes to a system in order to increase its capacity. This can be done by increasing server resources such as RAM, CPU, or disk space on existing servers. Vertical scaling is the process of upgrading individual components within an application architecture without changing any other parts of it. An example of vertical scaling is upgrading the RAM on a server; horizontal scaling might involve adding more servers to handle additional load.

**Embedded Analytics should support Continuous Integration and Delivery**

Many DevOps teams have adopted a continuous integration and continuous delivery (CI/CD) processs where code changes are automatically tested and released as they are committed to the code repository.

For this to be efficient, the application logic and user interface of the analytics platform must be decoupled from its data storage layer. This allows developers to work on different parts of an application independently, without worrying about how changes in one part will affect other components. For example, the user interface may see more development and incremental changes than the underlying data storage layer.

Containers and the alternatives

Traditionally all software ran on physical servers, but in the late 90’s Virtual Machines (VMs) as an alternative to running multiple physical computers. VMs enabled IT teams to reduce their costs by consolidating hardware and software resources into a single physical while also providing some flexibility by running different operating systems on the same hardware. However, VMs can be slow to start up due to having to boot their entire OS each time they’re launched.

More recently software containers have become popular as a lighter-weight alternative to VMs. Containers are a form of operating system virtualization that allow developers to package applications, their dependencies, and libraries into a single unit, called a container. Containers are isolated from the host environment and provide uniformity, portability, and scalability across different operating systems. Containers launch much faster and with less memory consumption than VMs and can be easily deployed across different infrastructures.

**Docker** is a popular container platform that creates and manages containers. It uses images, which are pre-configured packages of software that can be used to quickly deploy applications. It also has built-in orchestration tools for managing multiple containers.

**Kubernetes** is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. It is used to run, manage, and scale distributed applications, and works with Docker, Google Cloud Platform, and other container technologies. Kubernetes simplifies the process of deploying and managing containerized workloads.

Virtual Machines still have some advantages. For example, they provide better isolation between different environments as each VM has its own OS and application stack which can't be accessed by other machines or applications running on the same physical hardware - something that's not possible with Containers due their shared nature. Also, if you need access legacy systems then using Virtual Machines may still be best option since there often isn’t support available containerized versions these types of software yet.

# Security for embedded analytics

When designing our embedded solution, security must be the first priority. Data is at the core of analytics and should always be protected accordingly. To ensure successful administration, it is recommended to use existing security infrastructure and controls provided by the host application rather than creating new ones specifically for analytics. This simplifies management while also preventing potential openings in safety which could lead to a breach in security.

## Security priorities for an embedded analytics solution.

Successful embedded analytics solutions will prioritize the following security considerations:

### Adaptable options

It is important to ensure that your enterprise's security solutions are equipped with the necessary features for multiple scenarios such as operational reports and self-service analytics. Although initially you may only need embedded analytics capabilities, having options available in case of future success will be beneficial. For those selling shrink-wrapped applications featuring analytical functions, investing now in long-term flexibility will prove advantageous down the line.

### Permissions

Administrators should have the ability to set controls and permissions for either individual users or by roles, depending on the scenario. For example, role-based security is essential when embedded analytics are being accessed by multiple users in operational applications; however, if more personalization is desired (e.g., management or executive dashboards), it may be necessary to assign specific user permissions that do not match traditional organizational roles - such as giving a head of sales access to reports from marketing which other members of sales cannot view.

### Multi-tenant support

When implementing an embedded analytics solution in a multi-tenant host application, it is essential to consider how security, user and resource management are handled for each tenant. Not only should control of users and resources between tenants be managed securely but also other additional resources such as templates for dashboard or report design, integrated documentation and customization options. It is important that these secondary resources are stored safely; if they are not secured within the database by the tenant host then appropriate measures must be taken to ensure they cannot cross over into another tenant's environment. Even though this may seem unimportant in single-tenancy environments, when working with multiple tenants all data needs to be treated as confidential company information regardless of its perceived importance.

### Integration with 3rd party standards

A solution that can provide robust support should adhere to standards like Lightweight Directory Access Protocol (LDAP) and integrate with directory services, such as Microsoft Active Directory. This integration allows user authentication data and other security information, including passwords, to be securely shared across the network between various devices or applications connected in an enterprise system. Furthermore, role-based access control can easily be implemented since roles are already defined within the Active Directory itself.

### Stand-alone security model

In some cases, you may need to create and implement a standalone security model. In these situations, the embedded technology must have a suitable internal security system to meet your standards and a comprehensive API which is well-documented. With these priorities in mind, let’s look at different approaches to security that may be useful.

## Open and closed systems

A truly Open System would allow all users access to view and modify any shared content. This could be convenient for small teams or companies that have an open policy with data in general. However due to the potential dangers associated with such unrestricted sharing, we cannot recommend this approach.

A better method is the Restricted Open System where access to shared content is restricted in some way, either so that only certain people can edit certain content, or so that certain content is entirely invisible to particular people. An open system with restrictions is recommended for medium-sized or larger teams and companies, highly diversified user bases where reports aren’t relevant to everybody, or companies that want content to be viewable by everybody but editable by only a few.

For medium-sized teams and companies a Restricted Open System is recommended. This system restricts access to shared content in some way – making certain people able to edit particular pieces of content while others may not be able view the same information at all. Additionally, if you want your content accessible for viewing by everybody but only editable by a select few then this type of open system with restrictions would work.

Most large enterprises and nearly all multi-tenant platforms use some variation of a Closed System. In a closed system, administrators can only view and manage user accounts where they share the same user security group. This restriction also applies to groups of which they are members; admins cannot access any other users or groups outside their own. Consequently, there is no longer an All Users group on such systems – meaning it's impossible for all users in the system to be referred to as one collective entity.

If you are providing services to multiple tenants or companies, it is essential that you establish distinct groups for each one. We highly advise utilizing a closed system in cases where customers have clients from different businesses and organizations who should remain unaware of the existence of other users on the platform.

Having decided on your security approach, there are several ways in which you can configure how a user signs on to the operational application and the embedded analytics. The most common - and recommended – method is Single Sign On (SSO.)

## Single-sign-on (SSO)

Single Sign On (SSO) is an authentication process that allows a user to access multiple applications or websites with one set of login credentials. SSO reduces the need for users to remember and manage different usernames and passwords, making it easier for them log into their accounts from any device without having to manually enter their details each time they want access.

When deploying Embedded Analytics, Single sign-on (SSO) support is essential to enable a seamless experience for users, with no special login required for the analytic features. Instead, users will be authenticated through your host application.

SSO embedding works by adding a login button to an application (or adding a function to an existing login button) that enables users to log in using their SSO credentials. When the user clicks on this button, they are redirected to the designated Identity Provider (IdP) page where they can enter their username and password once before being granted access into both the host and embedded systems simultaneously without having manually inputting any additional details again for each one.

For a secure, standard-based way to exchange authentication and authorization data between an identity provider and a service provider, most developers use Security Assertion Markup Language (SAML.) The embedded analytic system will usually act as the Service Provider, and the operational system which verifies user identity will be the Identity Provider.

This process of consolidating identities and authentication methods is called Federation. There are two common ways to use federation for Embedded Analytics:

**Service Provider-initiated flow**

In a Service Provider-initiated (SP-Initiated) flow when a user attempts to access protected resources they are redirected by that site directly to the Identity Provider’s login page. This method allows users who already have accounts in both systems to quickly log into applications without having multiple sets of usernames and passwords for each system.

In Embedded Analytics, where the analytic system is the Service Provider and the host application the Identity Provider, this scenario is most useful when the business has some standalone analytic applications but also uses the same vendor’s features for embedding.

**Identity Provider-initiated flow**

In this scenario, Identity Provider-Initiated (IDP-initiated) flow, when a user attempts to access protected resources from the host application, they are redirected by that site directly to the Service Providers login where they could enter credentials and authenticate with SAML.

Even better for the user, an identity provider can use an API to login to the Service Provider for the user with all the necessary information (including signed security tokens) needed for them authenticate via SAML on the service provider side. This is preferred because it feels more seamless to the user and enables the embedded analytics to feel like a complete component of the overall system.

In general, look for a system which works with Identity Providers who support SAML 2.0 such as: Okta and Auth0 or Active Directory Federation Services (AD FS). Some enterprise applications such as Salesforce have their own identity providers that enable your users to use their Salesforce logins to interact with your embedded systems.

## Summary of security practices

Security should always be taken seriously to ensure both the safety of your customer’s information and their regulatory compliance. Here are some key points to keep in mind:

* If you provide analytics to customers or partners, you should use a separate instance of the analytics server for this. This ensures that your own systems are isolated from external users.
* Take the same approach for data. The data connected to your analytics server should not include any data that external users must not access.
* Enable a closed system.
* Choose an embedded architecture that supports SAML 2.0 and use strong authentication for your embedded instances such as Auth0, Okta, or AD FS .
* Use the Identity Provider-initiated scenario for SSO with an API for authentication

# Some other administration considerations

Deployment and security will be your primary administrative concerns, but there are some other administrative features to consider.

## Scheduling

It is rarely efficient, or, if running in the cloud, cost-effective to update an operational report or dashboard every time a user accesses it. This is especially true if the object is embedded in a screen which is always open on the user’s desktop. You will need the ability to refresh reports on a schedule or in response to event triggers from the host application. Integration with an existing enterprise scheduler, or a scheduler built into the database platform, may be useful if the enterprise already uses scheduled jobs extensively.

## Version management

If you are using DevOps processes, you will most likely integrate your embedded analytics with your existing version control system. Otherwise, you should ensure the embedded technology includes a built-in version management system. This may be critical for organizations with specific compliance standards which demand strict versioning of generated reports, such as in healthcare and financial services, where administrators need to guarantee accuracy of records over time so any changes are tracked, monitored and easily reverted if necessary.

## Report bursting

Report bursting as shown in Figure 6.2 is a process of distributing pre-formatted reports to different recipients. It usually involves splitting the report into multiple parts, each containing specific data related to that recipient's needs and interests.

Report bursting allows admins to customize individual versions based on criteria such as company name, department and so on, meaning that each report recipient receives the correct version with their own specific information already filled out within it.

Bursting is often used for invoice generation, financial reporting (creating a master report which is burst into individual cost center reports for managers) or even pay slips for employees.

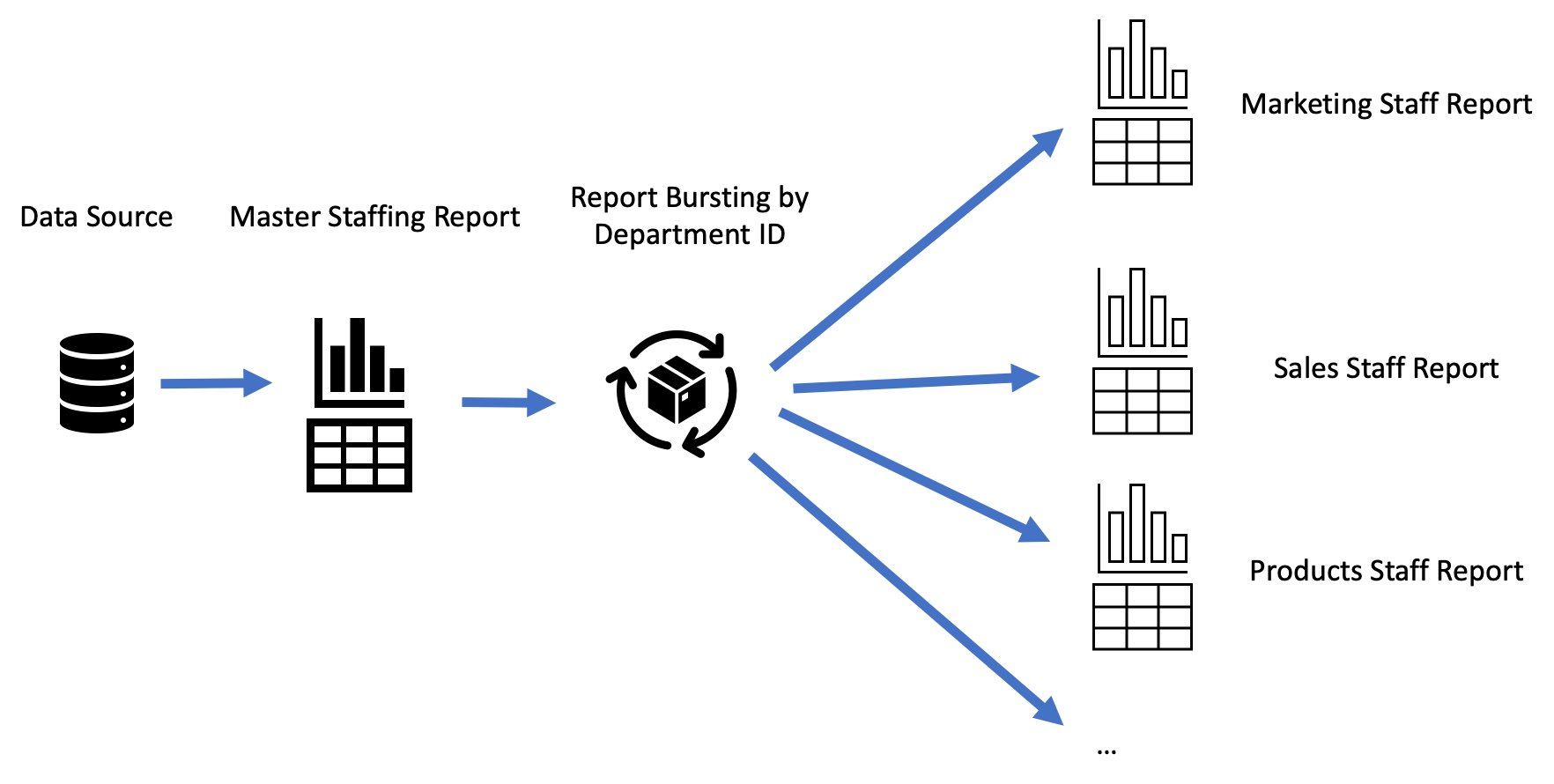


Figure 6.2 Report Bursting

## The administrative console

A good analytics solution should provide a centralized user interface for administrators to manage server resources, clustering, schedulers, security configurations and all other tasks they are responsible for. This unified portal or console offers admins an efficient way of viewing the allocation and utilization of their resources - not just making it more convenient but also increasingly necessary in order to ensure proper governance over our systems and understand how our assets are being utilized. Common tasks that can be completed from a console may include

* Configuring the embedded analytics Web Server including connection information and security certificate.
* Configuring logging, whether built in or using a logging service.
* International and localized settings such as language, currency, date formats and perhaps character encoding.
* Integrating with Active Directory or other directory service.
* Managing users and user groups including creating accounts, changing privileges and group membership.
* Scheduling refreshes of data.
* Setting the location of any temporary storage.
* Configuring or deleting artifacts created by self-service users.
* Monitoring server database performance and resource usage.
* Cancelling long-running or “stuck” queries that are consuming resources.
* Backing up (and restoring) data for recovery purposes .

In the next chapter we will look at data governance in more detail as it has become a significant consideration – if not a headache – to data administrators and developers. We will see that embedded analytics share many of the governance concerns of stand-alone application with a few unique requirements of their own.

7

Governance and compliance

Almost every day we read of data breaches, the egregious misuse of customer information, or some new hacking scandal. In this treacherous environment, any company using data extensively must grapple with ever more regulation and acute consumer pressure. As a result, the security and privacy of their data is top of mind for both anxious executives and hard-pressed IT teams.

Data governance as a practice has grown over the years from relatively simple policies for security and privacy to a broad, enterprise-wide initiative covering not only access to data, but the uses of data in analytics and machine learning and even the purposes of data for marketing, sales, research and so on.

In Chapter 6 we considered some of the particular security questions that embedded analytics raises. These questions are potent because we are combining two particular kinds of data: operational data in the host application and analytic data that has been integrated, cleansed, and often denormalized to enable efficient analysis. Just as analytics can be a valuable source of insight for business, it can also be a goldmine for less welcome use cases if it’s not rigorously protected.

In this chapter we will look beyond the security issues to include the demands of privacy, governance and compliance. We should start with some definitions …

The Pressure for Good Governance

Consumers are certainly becoming more aware—and more concerned—about the privacy and security of their data, especially with cloud services and storage. Software vendors and service providers, on the whole, work very hard to address these concerns. Their first motivation may well be the demands of new regulations and the possibility of stringent sanctions. However, along with the legal stick, we should consider the carrot of competitive advantage. A vendor who respects the privacy and consent of users can win new customers and enjoy greater loyalty than ever before from their existing community. Well-secured data also enables operational efficiencies within the enterprise, from onboarding new users to provisioning data and even enabling a more data-driven partner ecosystem. In short, privacy and security are not a burden for you to carry, but a commitment that benefits you and your customers.

# Governance, compliance, security and privacy

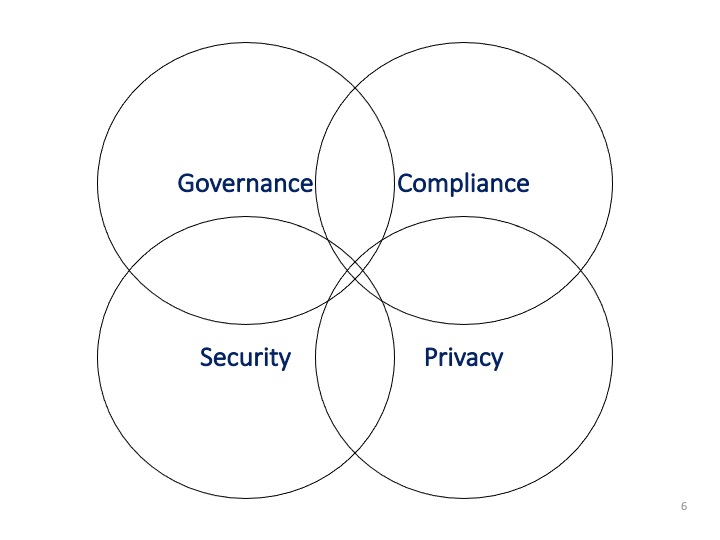


Figure -. The fields of data oversight and assurance

These four terms are closely related. All are concerned with some aspect of the safety and assurance of our data, and the oversight of our processes, but we need to be clear that they are also quite different. These differences require us to take a variety of approaches to ensure we are covering all the possibilities of risk.

To be free from threat and to be free from unwanted attention or intrusion are basic human needs which, nevertheless, vary widely in their terms between different societies and social contexts. Even when discussing data security and privacy in the context of regulations there are important, but subtle, differences between different regulatory regimes. So, the specific definitions that apply in your circumstances will vary, but it is still useful to have some principles in mind to guide your thinking.

## Privacy and Security

Security is about protecting data from unauthorized access.

Privacy is about appropriate use of personal data.

If we take privacy to mean that freedom from unwanted attention or intrusion mentioned earlier, and security the need to be free from threat, then a simple analogy may make the important differences clear.

In your home, if your blinds are open and anyone can look in, you have little privacy, even though with the doors and windows locked, you feel secure. However, if your blinds are closed but the door unlocked, despite your sense of privacy, which no passer-by can casually breach, you are not secure. And indeed, if someone enters your unlocked door, your privacy as well as your security have been compromised.

In terms of data, think of the controls you have in place to ensure the security of your network, such as firewalls and logins. You may be secure from unauthorized access, but if you have not taken care to ensure that even authenticated users do not casually see what they are not meant to see, you are not protecting privacy.

On the other hand, perhaps you have put in place protections to ensure that only users with the right permissions see certain data. You have done your best to protect privacy in that context. Yet, if the network is vulnerable, unauthorized users may still insinuate themselves; if they acquire the right privileges, both security and privacy are breached.

It is very important to understand, in any one scenario, what you are protecting: security or privacy? The answer is often both, but you will still find it best to consider them as separate scenarios and address them with specific solutions.

## Governance and Compliance

If your governance program is motivated in the first place by the need to keep on the right side of regulation, then compliance will likely be your first concern. Understand, however, that while governance and compliance are closely related, we must make some important distinctions.

Governance is the process of ensuring that you have sound processes in place and that these processes are diligently followed.

Governance is not about making better decisions.

Governance is about making decisions the right way.

This is quite a paradox for some. But I think it is realistic. We have been focussing on decision support and how to make better decisions. A good process should enable better decisions but cannot guarantee it.

If you think about civic governance, for example local government, you can see the difference quite clearly. You may not agree with the decisions your local government make, for example approving a new building or building a new traffic island, but that is quite different from believing that they have arrived at those decisions through a corrupt or invalid process.

Compliance depends on meeting a pre-defined set of criteria. These requirements may be external (laws, contracts, etc) or internal (such as codes of practice.) You can think of compliance as a check list - do you meet this set of requirements, whether issued by a government, a professional body or your own code of practice? You will be compliant, or not.

Nevertheless, it is difficult to consistently comply with regulations if you are not well governed. Without a thorough policy, broken processes can more likely lurk somewhere in your organization or architecture. Still, even a well-governed organization may not be fully compliant. Why? Because you could have overlooked or misunderstood some aspect of the rules, or not kept up with changes. Your internal processes, however carefully run, miss the target.

# Policies and practices

A full review of data governance best practices is beyond the scope of this book: it really is a specialized area. But we can provide some simple guidelines – some rules of thumb – which are relevant to the kinds of governance and compliance issues which crop up when embedding analytics.

From Gatekeepers to Shopkeepers

When describing new patterns of ownership for data and analytics, I have found it helpful to describe IT’s role - and IT’s mindset - as moving from being gatekeepers to shopkeepers.

What I mean by this, is that the defensive approach no longer works so well. Modern business intelligence strategy nearly always includes a strong element of self-service. Increasingly, users have skills and tools to do their own analysis, but still they need data.

Traditionally IT teams saw themselves as gatekeepers of the enterprise system, effectively keeping unwanted intruders out of important data sources. However, this model is typically too restrictive for all but the most regulated of new scenarios. Indeed, there is a real danger that the process is so constrained that even well-intentioned business users work around the system rather than within it, creating so-called shadow systems. These unsanctioned pools of data may be a considerable threat to governance in themselves.

It can be better for IT to think of their roles less as gatekeepers and more like shopkeepers. The shopkeeper is not unregulated (think of tobacco, alcohol or lottery sales) but does ensure that products (or in our case, data) are fit for purpose, appropriately packaged and available as needed. The advantage of this approach is that well-provisioned users (customers in the data shop) get what they need in a well-governed form, without the danger of being over-privileged with access to significant data sources. In short, rather than letting users into a system to explore, you push the appropriate data - and only the appropriate data - out to the user.

## If compliance is critical, you need a compliance team

Many of us work in highly regulated sectors - in public service, healthcare, finance and so on. It is important, therefore, to remember my point that governance does not guarantee compliance but is necessary to achieve consistency and confidence. So, I strongly suggest, as part of a governance program, that you set up a virtual team tasked with keeping on top of compliance issues. They should continuously re-evaluate legislation and identify where you need more, better or new coverage with your policies. They should also look into incidents and issues and track the numbers.

Not taking compliance for granted is a good practice in itself, but it also helps to relieve some burden and anxiety from others. It’s a big responsibility for a database administrator, for example, to keep on top of healthcare data-handling legislation and keep your systems running. Good policies frame the path to compliance, they are not barriers to business.

## Do not take privacy on trust

Consumers and customers world-wide increasingly worry about data privacy and, with good reason, don’t have much trust in enterprises to have their interests at heart. There are steps you can take to improve their confidence your own business.

* Be straightforward about your privacy policies and give customers control We are all familiar now with websites which ask us to specify our cookie policy. You can - and should - ask customers to allow for other uses of other data you may have about them, such as demographics for research, usage of the system for product improvements, trends for marketing.
* Put policies in place at every level of the enterprise which enforce these preferences as rules. Some rules can be applied in code or technical procedures - check that you are not extracting demographic data from a customer record into the warehouse unless the customer has given permission. Other rules - such as, *don’t use these customer records for product research* - may be more like policies. Either way, your governance program needs to cover them.

## Look for secondary benefits of good governance

So far, these practices have been focussed on what you may think of as the restrictive aspects of governance - trying not to do the wrong thing. Good governance also enables a lot of good to happen.

A well-governed improves access to data and encourages efficient re-use of analytics and reports which have already been created. This works, because policies define in advance what data is relevant and permissible to a role and therefore it can be confidently provisioned. Poorly governed systems tend to raise a stream of ad-hoc requests for data access which are disruptive for IT and prone to error including the compliance risk of over-provisioning permissions just to get the job done.

Although better decisions are not the direct goal of governance, a well-governed decision is likely to be more collaborative, better understood and more widely supported than otherwise. Confidence in the process lends confidence to the decision, especially if the policies have been crafted as a partnership between teams rather than assembled along departmental lines.

Collaboration can improve in a well-governed organization. Many CIOs complain about data silos in their business, or data hoarding by departments and individuals. These are real problems often caused by data owners being unsure if users outside their span of control will handle data responsibly.  With the right policies in place, data owners should feel more comfortable that sharing and collaboration will be accountable.

## Commit to openness, awareness and training

Several of these practices build on a shared understanding of the governance process across your organization. If data users don’t know about the program it cannot be effective.

* Be open about the program, its goals and its measures of success. Publish the strategy, describe the processes, share the metrics.
* Awareness of the program should form part of all onboarding. If you have compliance training for other areas - such as non-harassment - work with those teams and HR to get data governance onto a similar track.
* In all technical training related to data, such as for business intelligence tools, include relevant elements of the governance strategy and how it is specifically realised in the tools and platforms under review.
* Training also needs to cover the importance of data quality - and how policies help - along with the availability and reuse of governed data sources. Data analysts and report authors in particular, should not feel they are trading agility for policy. Rather they should see governed data as a resource opening up possibilities for more assertive work.
* It can prove useful to implement a “governance assured” stamp of approval which you can add to dashboards, reports and other artefacts to show that the data in question has been governed properly.

# Governing your governance

An effective governance program is a continuous effort. As an organization you will adopt new technologies and platforms and will need to work out how program can govern them or how the program should be extended. New roles will emerge. Regulations will change.

It should be clear, then, that you need to repeatedly evaluate your policies. How often? Certainly, an annual assessment makes sense at a minimum, because over that period a lot can change. Other reviews will be ad-hoc, for example when completing a merger and acquisition which brings new data, new people and new tools onboard.

Some sectors, such as financial services, may see frequent changes as not only data legislation changes, but also rules concerning money-laundering, sanctions, liquidity and credit and so on.

You need to find this rhythm of review for your own business. That, in itself, can be an eye-opening exercise.

One concern may be very relevant to you in a large organization, or one with very strict governance needs. Where does governance sit in the overall hierarchy? Does the governance team report to the CTO or CIO? Or perhaps to a Chief Security Officer? Or a Chief Compliance Officer. I have seen many different configurations. Some did not work because the specific fit of individuals to roles was not good. Others worked well because there was a good commitment all round to make the program successful.

I have suggested security is related to governance but also a separate, specialised domain. I hold to that view - it is too technical a field for an effective CSO to also carry the weight of business process management.

Compliance is also related to governance. So a Chief Compliance Officer may well manage a governance team, directly or virtually.

Finally, if neither of these roles exist, I would prefer to see governance reporting to the CIO than to the CTO. I often find I need to emphasize that governance is not a technology problem with a technical solution. Rather it emerges as an issue of people, processes and technologies working together and the office of the CIO may be the best fit.

Governance is journey, no doubt and not an easy path, especially as you get started. However, a well-governed data infrastructure runs smoothly for business, IT, customers and partners. It is a worthwhile investment.

## A security and privacy virtual team

With an understanding that security and privacy are separate but associated subject areas, you’ll find it practical to establish two separate but associated programs, even if these programs are run by one virtual team.

The focus of a privacy program should be on the appropriate legislation that determines processing, protection, and retention requirements. The program must also take into account consumer expectations. Realistically, compliance with regulations will be top of the agenda. If regulations in your field are complex - for example, in Healthcare - you may need a legal specialist on the privacy team, perhaps from your office of General Counsel.

The requirements identified by the privacy program are passed to the security program for implementation. However, don’t specify the specific processes or technology to be implemented at this stage.

It’s the security team who has the knowledge and capability to set up suitable protections and controls to meet the needs identified by the privacy program. An important advantage of keeping the two separated in this way is that the interpretation of regulations won’t be unduly influenced by concerns about technical limitations. As a result, when the security team reviews the unbiased requirements from the privacy program, it’s possible they’ll identify technical demands that require new, or different, tools and platforms.

## Governance in the cloud

As we saw in Chapter 6, it’s a sound practice to define and implement security and privacy controls at the lowest practical tier—as close to your data storage as possible. Benefits will bubble up from there. On the cloud, however, there’s another consideration. You should look to the underlying cloud platform itself to provide excellent security and privacy features. The more support the platform provides, the less you have to worry about and the more consistently you can apply your best practices.

With this support from the platform, an enterprise deploying analytics in the cloud, will find security less daunting. Encryption, for example, is a critical feature for both security and privacy. Look for comprehensive encryption features in your data storage platform. You should not have to design and implement such demanding features yourself when your cloud provider has greater expertise and resources to make them available as part of your subscription.

In reality, securing the entire cloud infrastructure is not a role for the analytics administrator or developer. Nevertheless it is useful to know the basics.

## Business continuity is a security and privacy issue

It’s one thing to protect your data when your systems are running smoothly; but you also have to consider its resiliency and availability in the event of an adverse incident. For customers and users, it’s infuriating enough when a system is down. They want service restored quickly, and not just because they want to get back to work. In the darkness of downtime, they can’t help but worry that their interests have been compromised: the wait is too often a protracted agony, relieved only when the application is back up and running. Think of how you feel when you can’t access your bank accounts online: it is not just an inconvenience, but a real worry.

Some consumer-focused legislation, such as the European Union’s GDPR, explicitly refers to this critical need by requiring that the organization “implement appropriate technical and organizational measures” that include “the ability to restore the availability and access to personal data in a timely manner in the event of a physical or technical incident.”

Cloud services are typically very robust and reliable, but outages do happen. It is critical to ensure your cloud service not only automatically performs regular database backups but also guarantees *Point-in-Time Restore* from these backups.

# Developing a governance strategy

Some companies worry that governance will be disruptive to their work, or a burden that holds us back, or an expense that you must budget for. But the simple truth is, you are already doing some data governance right now and you can develop a strategy starting from here. In fact, it is best to do just that.

Today you may not have a documented data governance policy, but you certainly have people who manage your data: a database administrator who sets permissions for access, IT staff who diligently backup and restore, a network manager who at least checks that your business intelligence tools are licensed properly.

To start where you are, create a directory of the data assets that your organization works with. At the same time, add a a listing of who exercises any responsibilities for that data - whether they are an administrator, manager or just an engineer ploughing a lonely furrow - and find out who they report to.

In the course of this investigation, you will likely find some sobering, if not shocking, oversights and gaps in coverage that need quick attention. But at the end you will have a simplified catalog of assets, roles and accountabilities. It’s going to be informal, but it will reflect the messy reality of your current governance.

This informal state of affairs is no doubt a little messy because it is unplanned. But it is reality - because it records your present-day organization and needs.

It is too much to ask, and too risky, to remodel your organization in order to better govern your data. Instead, construct some formal scaffolding around your existing ad-hoc structures. You can do that by identifying some of the key roles involved.

Who advises or is consulted on what the governance controls should be? We could include business users, the office of general counsel, IT teams - it is best at first to cast the net widely. You will soon learn who has useful insight to share.

Who implements the controls? Some aspects of this work can be quite technical, for example applying row-and-column level permissions in a database. Other times it may be more administrative: HR ensuring a new hire is assigned the right job title into inherit the right permissions automatically.

## Measuring the success of governance

I have already suggested that data governance strategies may not have a measurable impact on the “bottom line” of business. But I also mentioned the heavy fines that regulators can impose, not to mention the cost of restitution and repetitional risk if things go wrong. So, we can certainly say that bad governance - or none at all - may indeed have a financial impact.

Nevertheless, “not getting fined” is not a great KPI for your annual review. How else might we measure the success of a governance program?

One way is to specifically identify data-related issues and incidents as part of your existing IT issue management system, but also ensuring that all data-related incidents are reviewed for governance implications. For example, a user losing access to a database may be a simple incident, but if it is resolved by elevating their permissions to give them access to more data than they strictly need, that would be a governance issue.

Recording, investigating and tracking such cases will quickly grow a body of knowledge and associated metrics about how governance is improving - or degrading - over time.

In addition to incidents, which are by their nature exceptions to the regular running of your systems, it is useful to measure the overall performance of your system. For example, simply tracking how many users have access to a system, with what permissions, and how much they use it gives you at first a useful benchmark and, over time, a growing insight to how data is being used.

You may not, at first, have a good sense of whether a metric is improving or not as it changes, but the improved insight will help. For example, if you are adding more users that may be because of increased data literacy - good. But if those are users are creating more copies of the data in more reports without reuse that would suggest poor governance.

In short, tracking incidents and issues is crucial to your success. Measures of insight and oversight develop over time and can even be useful used as benchmarks once you understand what you are looking for.

# Summary

This has been a very brief overview of governance and compliance policy as it may affect embedded analytics. Nevertheless, there are some important ideas here.

The move from being gatekeepers who prevent data access to shopkeepers who enable it is a key element of embedding analytics. The purpose of the work is to bring data and insight to new users and new scenarios.

The distinctions between security, privacy, governance and compliance are useful to bear in mind when addressing issues, especially across team boundaries. The simple questions … Is this a governance or compliance issue? Privacy or security? These can unlock important and insightful conversations.

Finally, you can begin to develop a data governance strategy even from a seemingly unpromising and informal starting point. The trick is to get started. It is much easier to improve governance gradually than it is to handle the problems that can arise from a poorly governed infrastructure.

## Governance going forward

We have seen one reason why governance is so important to both consumers and businesses: a growing concern with security and privacy of data. This concern is keenly felt when decisions are made, not by human beings, but my automated systems.

Automation of decisions has an important role to play in the future of embedded analytics. In Chapter 9, we will look at some scenarios and techniques where machine intelligence and decision-making can be combined with embedded analytics.

8

Beyond the Spreadsheet

Spreadsheets have an uneasy co-existence with visual analytics solutions. Most visual analytics professionals will have plentiful stories about how difficult it can be to drive adoption of visual analytics in environments where spreadsheets are considered the default option for working with data. No matter how sophisticated and interactive a visual experience is, the cry from users in the real world is often simply “can I get this in Excel?”

However, over the last few years spreadsheet tools have developed at a pace comparable to visual analytics, with new paradigms and features that allow users to use spreadsheet tools as the heart of accessible, collaborative and programmable *platforms* centeredaround their data. Some of these tools are driving the development of entirely new analytics approaches, some eschew the entire concept of visual analytics in favor of grids and tables, and some are purely designed to democratize the process of collecting and managing a SSOT (single source of truth) for data within an organization.

So, how should we consider spreadsheets in relation to embedded analytics? In this chapter we’ll explore some of the history behind spreadsheets, see how the product landscape has evolved over the last few years, and bring to life ways in which modern spreadsheet tools can add real value to an embedded analytics solution.

In order to help support conversations around tools you may have within your own organization, we’ll also attempt to answer the question of whether spreadsheet tools can be considered as analytics platforms in their own right.

# Setting the stage

For the purposes of this chapter, let's set out some of our own loose definitions to help us differentiate terms that are often used interchangeably in the real world.

* **Tables.** The typical way in which data is represented and interacted with in a spreadsheet tool. We're using this term to encompass the most basic examples where data is organized into rows which share common columns, and more complex examples, like pivot tables. Synonyms might include terms like "sheets" or "grids".
* **Raw data.** The source of data used by a spreadsheet tool. This might be as simple as row-level, disaggregated data that's been pasted into a sheet in your spreadsheet tool. It could also be data stored in a relational database that the spreadsheet tool connects to.
* **Delimited data.** This term encompasses a variety of file formats that are used to encode and exchange data generated by databases, ERP, CRM and spreadsheet tools. Data in a delimited text file is arranged in rows, with a "delimiter" character used to separate the values in each row into their respective columns. Different characters can be used as delimiters - you'll probably be familiar with the most common example which uses a comma to separate values - a CSV file.

As much as delimited data is a primary way of exchanging data between systems, whether an analytics solution provides options to export delimited data from a system is a contentious point when trying to promote adoption, governance and best-practice administration – a point we’ll cover later.

* **Spreadsheet tool.** The software that allows you to create and manipulate raw data and tables. Again, we're going to use this term to encompass ubiquitous software like Excel along with more modern derivates that leverage similar functionality to provide SaaS-based platforms with broad features, like Airtable and Google Sheets.

# Let there be spreadsheets

Over the last two hundred thousand years, humans have evolved to be pattern-recognition and "visual enumeration" machines. Our peripheral vision and visual acuity are critical traits that enabled early humans to evade predators, hunt and make critical decisions about the environment around them. Our whole view of the world is informed by these traits.

So, what does this have to do with embedded analytics and spreadsheets? Clearly, it's a bit of stretch to compare and contrast an ancient human evading predators with someone sitting in an air-conditioned office reviewing their latest marketing reports; but those traits are still present, and some of the ways we see and interpret data still tap into these ancient behaviors. Consider some examples:



Figure 8-: Ripening apples on a tree

Figure 8-1 illustrates a great example of **selection**. It’s important (even potentially life-saving) for us to be able to recognize ripe fruits, so we’re able to very quickly identify common visual traits – a change of color, in this case – and group those elements together as a single selection that we’re interested in. In this case, we’re interested in picking ripe apples, but consider a more relevant example:

Table

Description automatically generated

Figure 8-: Selection as applied to tabular data

Figure 8-2 shows us how this selection trait can apply to a more modern (but perhaps less tasty) example. Highlighting rows in data according to a condition leverages our ability to quickly identify the rows we’re interested in. Notice how it’s intuitive to scan through the list of sales reps and identify the names of four individuals. This is your visual selection trait in action.

A group of wild animals in a field

Description automatically generated with low confidence

Figure 8-: A clan of hyenas

**Enumeration** is another key trait tied to our visual acuity. Figure 8-3 shows a potentially dangerous situation for an ancient human; a clan of hyenas looking particularly curious. How do we evaluate this danger? One hyena is very unlikely to present a serious threat, but a group – certainly one as large as this - is something to be worried about. There’s a nuance here: we’re not counting the individuals in the group; we just instinctively know that a group of this size is dangerous.

Fortunately, our modern world rarely involves such danger, but the trait is still useful.

Table

Description automatically generated

Figure 8-: Enumerating quantities

Using the same example set of data, figure 8-4 shows us how we can enumerate quantities very quickly, without explicitly counting values. Most readers would be able to instantly determine that there are more Pencils than Binders in the selected column, just by virtue of some basic highlighting.

So, we can demonstrate that these traits are very well suited to interpreting data arranged in a table, but one particular trait was arguably the genesis for the *very first* use of written tables in history: pattern recognition. We are pattern recognition machines; able to find structure and repeating visual motifs in the world around us instinctively – outperforming even the most sophisticated algorithms.

Footprints in the sand

Description automatically generated with medium confidence

Figure 8-: Footprints in sand

Figure 8-5 shows a pattern we’re all instantly familiar with, footprints in sand. Subconsciously we’re determining who left the prints, and – importantly – where they lead. They also have *direction*, which gives us even more rich information about the immediate surroundings. How does this relate to interpretation of real-world data in our example?



Figure 8-: A section of the oldest-know multiplication table

Figure 8-6 is possibly the very first use of a “table” to represent data for interpretation. It’s a multiplication table believed to be around 2,500 years old. It relies on directional pattern recognition to allow a viewer to select multipliers from the uppermost row and required column, follow the intersection between them and determine the product.

The full version of this table allows the multiplication of any whole or half integers between 0.5 and 99.5. It’s a fascinating example of how tables evolved to solve real-world problems, and a use-case that’s still in use across the world today, on blackboards, books, and – you guessed it, Excel.

# Are spreadsheets analytics platforms?

Written tables stemming from the multiplication table example above were the first way humans were able to systematically and rigorously interpret data - which is a solid definition of "analytics" as we know it today.

However, it took the advent of personal computing in the late 70's to give us a glimpse of the truly transformative change that spreadsheets could bring about. The first commercially available spreadsheet tool, VisiCalc, (figure 8-7) was made available for the Apple II computer in 1979, and the impact was immediate. Manual bookkeeping processes, laborious calculations and error-prone repetition were eradicated with re-computable formulas, easily navigable tables, and consistent data entry methods. Finance teams adopted changes, their companies saved millions of dollars, and the concept of the modern spreadsheet was born.



Figure 8- : An example of an early spreadsheet tool

Finance teams weren't the only people taking advantage of this new technology. It wasn't just the process efficiency that spreadsheets delivered - the flexibility that spreadsheets provided was leveraged by a new audience of managers, human resources teams, supply chain analysts, marketing executives and almost every other function you can think of within a commercial organization.

Since then, the pace of change has been relentless and spreadsheet tools have become ubiquitous. If you've got a laptop, it's safe to say you've seen - if not used - a spreadsheet tool, and approximate estimates suggest that up to two billion people use spreadsheet tools like Excel and Google Sheets every day.

## The ubiquity of Excel

Very quickly, the spreadsheet tool landscape proliferated. SuperCalc, Lotus 1-2-3, Multiplan and many other vendors drove development throughout the eighties, and innovation happened fast. Users were able to write more complex calculations, the UI (user interface) became more sophisticated, and features were introduced to build charts and graphs natively within the application.

When Excel started to gain popularity in the late 1980’s, it laid the foundation as the most advanced spreadsheet tool available, a claim many people would still agree with well over 40 years later. Sticking with Excel for the moment; what does it do now that it didn’t do then? The list is significant, but some of the core features it provides now are:

* The ability to maintain multiple, independent and dynamically referenceable sheets within a single file
* A robust and extremely wide-ranging collection of functions, including a framework to write custom functions in VBA (Visual Basic) to extend functionality and integrate with other third-party systems
* Integration features to obtain data from a very wide variety of sources, like relational databases
* Sophisticated data modelling, query editing and transformation features
* Math and trigonometry functions, along with built-in statistical modelling and forecasting tools
* A large range of sophisticated and customizable charts and graphs
* Capable online versions to allow for collaboration and version control
* Lots of options for exported **delimited data** for use in other tools

If you’ve even evaluated an analytics platform, this would be a sensible list of criteria to use – so it’s clear that Excel *can* demonstrably fulfil the function of an analytics platform. In fact, thousands of organizations around the world are using Excel as their sole platform for data orchestration and analysis right now. Even if they don’t know it.

Excel is so ubiquitous and feature-rich that it presents a path of least resistance to organizations when faced with data and analytics requirements. In many cases, organizations have already paid for Excel as part of an existing Microsoft purchase, so they’ve already got the software – a big hurdle to overcome. Then, it provides a near-perfect fit for so many day-to-day challenges:

* *Where do we store our monthly finance records?* Excel isn’t necessarily a perfect solution for storing data, but with careful management it’s a great fit for archiving lots of row-level data for future analysis.
* *I need to import some third-party data I found on the web and prepare it for analysis.* Well, you’re already familiar with the way Excel works, so let’s just paste the data into a sheet and start working on it.
* *I must build a chart for the weekly sales presentation.* You can build the chart you need in a couple of clicks; along with drop-shadows and gradient fills, which everyone loves, right?
* *I need to do a quick and dirty budget balance for the next staff offsite meeting.* Excel’s here – why not just use that? Basic formulas are easy to write, and the file can be sent straight to your manager for approval.

The fact that Excel is such a good fit for these kinds of challenges has cemented its position as the default choice for a data and analytics platform in many organizations, but it’s not necessarily the *right* choice.

## What Excel doesn’t do

Excel is undoubtedly a great product, but the sheer scale of data challenges facing many organizations today means that a true analytics platform needs to provide far more robust controls over the way data is shared and consumed by users.

It’s worth noting that many A&M (acquisition and merger) strategies will specifically include audit-style exercises which require companies to formally demonstrate data and analytics capabilities in many of the following areas in order to be considered as “data mature”.

Here are a few ways in which Excel and similar spreadsheet tools don’t meet the requirements of a true analytics platform.

### Controlling the proliferation of content

Turner works in a large recycling plant. His daily responsibilities include downloading processing data from their plant management software, aggregating minute-level data to create KPIs, which he then sends to his manager. Every day, he saves a work in progress version on his local computer, then a final version. Sometimes, a *really* final version is created as he corrects mistakes. In any case, he also emails a copy to his manager, who saves this on his local computer.

Graphical user interface, text, application

Description automatically generated

Figure 8- : An example of duplicate reporting content

Figure 8-8 shows a somewhat contrived example of this. We’ve all done it, and it’s not just endemic to Excel. However, it does illustrate one of the major headaches of using completely democratized software to manage data – in Turner’s example, there are at least 4 or 5 files created *every* day (assuming neither Turner nor his manager distribute copies to anyone else) named in arbitrary ways, containing similar data. That’s over a thousand files a year, distributed across various local computers, network storage locations and internal mailboxes.

Then, imagine a scenario where an external auditor asks for the last years’ worth of data. Turner will have a *lot* of work to do. Even worse, imagine a scenario where an external auditor asks you to validate that a single source of accurate processing data is only stored in one, secure location. Turner’s manager now has a lot of work to do!

This is the problem with proliferation of content – once it’s permitted, it’s almost impossible to control and very difficult to refactor.

### Ability to support computation on large volumes of data

As of the time of writing, Excel can handle 1,048,576 rows and 16,384 columns in a given sheet. This – need it be said – is a *big* spreadsheet, but in the real world most users are likely to encounter performance problems with Excel with far fewer rows, especially once the data needs to be manipulated and formulas computed.

The general complexity and volume of data (extracted from CRM / ERP systems) is ever-increasing, and whilst having to manipulate or otherwise work with million-row data sets seems unlikely, it’s far more common that many organizations realize, especially when data is being manually integrated from multiple sources.

To handle the manipulation and transformation of datasets this large, you really need to look towards dedicated database solutions.

### Security and authorization

Potentially one of the most crucial aspects of any data and analytics solution, a scalable security and governance strategy is very difficult to implement reliably over the top of a tool like Excel.

We’ve talked in depth about what an effective governance and compliance solution looks like in the previous chapter, and if readers review that chapter with Excel in mind as the analytics platform, you’ll come to an obvious conclusion: almost none of the core governance principles can be effectively implemented if Excel is considered to be the sole analytics platform – and certainly not without leveraging many costly additional, external services to enforce even basic principles.

### Sharing and embedding

Last, but by no means least, a true analytics platform depends on the ability to share content with users in a way that doesn’t conflict or break any governance principles already established – or lead to content proliferation. A free-for-all distribution without any form of control is clearly undesirable, but Excel doesn’t have a centralized model for being able to share content outside of its own Microsoft-powered platform, and although it's possible to embed Excel documents hosted on third-party platforms, there’s no inherent ability to control of otherwise federate access to that content within Excel itself.

To put it another way, if you wanted to embed Excel content in your own website or application, you’re going to have challenges which simply don’t exist with many dedicated data and analytics platforms.

## Almost an answer

Hopefully you can see why the question of “are spreadsheets analytics platforms?” isn’t quite as straightforward as it seems, and your answer will undoubtedly vary depending on the way in which you currently use spreadsheets, and what your organization’s relationship with data will look like in future.

However, a reasonable statement to make would be as follows:

Spreadsheet tools aren’t inherently suited to be used as analytics platforms, but their level of adoption and powerful and well-understood features make them an integral part of many analytics solutions.

Fortunately, the world of spreadsheet tools is no longer limited to monolithic, client-based applications like Excel. New, highly focused, SaaS-based offerings are now bridging the gap between democratic access to data and tight governance in innovative ways. In addition, some tools have very interesting potential for embedded use-cases.

# Beyond Excel

As Excel has maintained its position as the most popular spreadsheet tool, the broader landscape of spreadsheet tools has hugely diversified. We’re now seeing predominantly SaaS-based products gain traction; some of which focus on addressing very specific business challenges, like project management, or survey capture.

SaaS generally provides more flexibility for integration, but of particular interest to us from an embedded analytics perspective is the use of these tools (and native **tables**) in a relatively new type of interactive collaborative development tool – the browser-based, computational notebook.

Let’s explore a few of these categories, both old and new:

## Simple reporting and analytics

Many vendors offer traditional spreadsheet tools that are geared towards supporting basic reporting analytics, and some of these tools tend to leverage the fact that legacy tools like Excel are likely to remain the *source* for existing reporting data.

For example, GRID (<https://grid.is/>) allows users to connect to existing data in Excel and then create rich, interactive visualizations from the data, which can be shared with other users and include narrative and contextual feedback, like comments and feedback. GRID does have its own native spreadsheet editor, but the product is primarily positioned to fulfill a very common use-case: the presentation of, and collaboration around analytics; something Excel does not natively provide.

Comparably, Google Sheets is a more generalized online spreadsheet tool, but also leverages Google’s robust app ecosystem to provide rich collaboration capabilities, not just within Sheets, but also across other products in Google’s Workspace like Slides and Docs.

Tools like these trade heavily on the users familiarity with traditional spreadsheet concepts, like writing calculations, formatting, and the general “flow” of working with data – copying and pasting content, created summary or aggregated tables, etc. As such, they’re often a appropriate choice if you’re considering embedded table content in your analytics solutions.

Table

Description automatically generated

Figure 8-: an example of an iframe in a webpage containing an embedded Google Sheet

Figure 8-9 uses the same table we used in our examples earlier in the chapter. In this case, we’ve created a Google Sheet to contain the data in question and embedded that using the native iframe-based sharing capabilities in Google Sheets. You can imagine how a webpage could easily be created that embeds several sheets containing both tables and charts, all potentially generated by the same tool.

This is a very simplistic example, with a publicly available Google Sheet. Recall that security and authorization are critical components of a true analytics platform and should be considered a default requirement if your use-case requires anything other than unrestricted access to content.

## Integration and collaboration

We’ve touched on the idea of collaboration being a potentially valuable feature for a spreadsheet tool. Some vendors have taken this concept much further to build spreadsheet-based products that use tables as a central repository for powerful collaboration and integration use-cases. This repeatable and re-usable “framework” approach provides flexibility – interfaces can be used to create very intuitive and visual ways for data to be captured, forms to be created, projects to be tracked and data to be analyzed.

Additionally, many of the tools in this category like Airtable (<https://www.airtable.com/>) have connectors for other common productivity tools, which can be used to move information between the tool and other services. What does this mean in practice?

* Content from other applications, like visual whiteboarding platforms, can be displayed alongside related tables in the tool.
* Notifications can be sent from the tool to communication platforms like Slack and Microsoft Teams when tables or user-facing interfaces are updated.
* CRM or ERP tools can be updated directly when statuses are updated in the tool.

The possibilities are endless, and tools in this category often present a compelling and cost-effective option for organizations seeking to solve operational challenges.

## Project management and workflow

Spreadsheet concepts lend themselves well to project management methodologies. Tasks with any number of attributes (duration, date, status, complexity, etc.) are a natural fit for tables, and cell formatting makes pseudo-Gantt charts easy to create and maintain. Vendors have sought to take advantage of this to create tools like Smartsheet (<https://www.smartsheet.com/>) and Monday.com (<https://monday.com/>) that provide a broad range of features to support typical management and workflow-based requirements like task management, lead management, onboarding, contact management and campaign management. It’s a rich seam to mine for vendors because the features to support all these use-cases can all be derived from the basic concept of storing data in the form of a spreadsheet.

## Computational Notebooks

Potentially one of the most interesting ways that table-based data can be used in an embedded analytics context is the relatively recent concept of the computational notebook. In basic terms, a computation notebook (or just “notebook” for the purposes of this chapter) is a web-based environment where documents can be created that contain text and rich media alongside defined blocks that can contain code, visual analytics, mathematic representation and – of course – tables.

Notebooks gained popularity because they very effectively combine core IDE (Integrated Development Environment) concepts like the ability to write, edit, debug and run code with collaborative and narrative concepts, like explaining what the code does (with text) and visualizing the results (with embedded tables and visual analytics). A notebook becomes a low-dependency, self-contained interactive environment for developing code that’s perfect for team-based code development, data science, and analytics.

A screenshot of a computer

Description automatically generated

Figure 8-: A sample computational notebook

Figure 8-10 shows a basic example of a notebook hosted on Google’s collaborative notebook platform, Colaboratory (<https://colab.research.google.com/>), a great resource for those who want to experiment with notebooks. Our notebook includes formatted commentary, snippets of code that can be executed in real time, and the results of the executed code: an embedded, interactive **table**.

Behind the scenes, a notebook is a document containing code, text and rich media which uses a computational component called the *kernel* to execute and code and integration processes that may be included in the document. It’s typical for the notebook platform and kernel to be hosted on a cloud platform provider like AWS or Azure, whereby the kernel can leverage scalable compute resources to handle the code execution. This makes notebooks a compelling solution for computationally intensive AI and ML projects, where *explaining what you’re doing* can be presented alongside code – and the results of that code - that can be executed in real-time.

One inevitable consequence of product diversification is that terminology and nomenclature become more complex. Vendors seek to differentiate their product features by introducing completely new terms that represent similar things, and a great deal of time in product evaluation projects can be spent simply learning what these new terms represent. It can feel like an unproductive effort, but it’s important to do in order to meaningfully compare features.

# Putting spreadsheets in context

Should you consider spreadsheet tools and tables as embeddable objects that can add value to your solution? Unequivocally, yes. But with several caveats:

### Choose the spreadsheet tool carefully

If you’ve decided that your embedded analytics solution requires tables for a legitimate reason, and you believe they will add value or improve the user experience, pay careful attention to the considerations we discussed at the start of the chapter. Embedding tables from traditional spreadsheet tools (like Excel) will potentially break defined governance principles, open up security risks, and be far more difficult to manage in comparison to embedded visual analytics objects served up from a dedicated visual analytics platform.

### Don’t break the paradigm of visual analytics

This might be a more diplomatic way of saying “don’t be lazy”. Visual analytics exist because they leverage those ancient selection, enumeration and pattern recognition traits to communicate insight to a viewer as effectively as possible. If you’re considering embedding a table, ask yourself – is this *definitively* the best way to communicate and present data for the purpose in question? If not, then reverting to an appropriate chart or graph (or KPI) may be a more appropriate choice, irrespective of the features and functionality the spreadsheet tool may provide.

### Pursue consistent methods for interaction where possible

In many cases, (as with the concept of computational notebooks) the mechanism or spreadsheet tool being used to generate the table is going to be different to the tool used to generate visual charts. For example, you may embed a ThoughtSpot liveboard in a given solution alongside a table generated by Airtable – two completely different software solutions. In this case, the methods for interacting with those embedded elements are – naturally – going to be different. Drilling down may require slightly different processes in each tool, and certain interaction methods, like right-clicking, may simply not work in both tools.

To an extent, this is unavoidable (and perhaps expected by users) but nonetheless it’s important to think very carefully about how users will manage and understand the interaction options available to them if multiple vendors are being used.

#### Consider highly-targeted use-cases for tables

At the start of this chapter, we briefly mentioned the common pattern for users who simply want to download or export **delimited data** from an embedded system. Again, there’s a case to be made here that appropriate use of visual analytics may negate the requirement for the export in the first place – after all, if the user wants to download the data in order to build a chart in Excel, why not build an embedded interactive version of that chart in the first place and keep the user in the system?

However, there are legitimate reasons for users needing to download data. They might have to conduct analysis which isn’t possible in the embedded system they’re using. They might be required to upload the data to a third-party system as part of their own workflow. In the case of external-facing, or monetized embedded analytics solutions, there may be a contractual requirement to make the data available.

In any of these cases, tables perform a valuable role in your embedded solution: previewing data for exporting and giving the user the ability to refine and otherwise control what eventually makes it into the downloaded file.

Graphical user interface, application

Description automatically generated

Figure 8-: A table presented to the user when downloading data from a Tableau view

Figure 8-11 shows a great example of this. Tableau can preview data used in a particular chart or view and download the underlying data (assuming the user has appropriate permissions). In this case, the user is shown a table representing the data they’re going to download, with the ability to preview a selected number of rows and show or hide columns that may not be used in the chart itself. This is a good user experience, and a perfect example of how tables can be used to add value to an embedded experience.

# In conclusion

Spreadsheet tools are no longer just for creating tables, and users have access to an enormous range of tools, platforms and nascent technology based around the concept of storing data in rows and columns. This doesn’t necessarily mean that tables are *needed* to form part of an effective embedded analytics solutions, but with careful consideration and tool selection, embedding tabular content to provide additional context, provide users with options they expect (like downloading and exporting data). Providing a guided analytics experience is something that any solution architect should explore.