



# Why Open Architectures Matter in BI

## Openness White Paper

April 2021

### Whiskey

Last 12 Months

TOTAL ORDERS

**3,976**

3,120 ↑ 12%


**\$2,941,956**

\$2,963,907 ↑ 8.9%

**73%**

Target: 3.26m

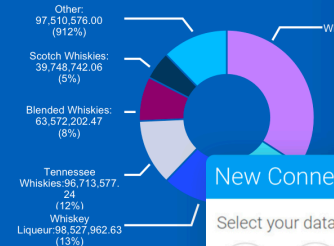
Performers

-  Canadian Club  
**43** ↑ 12%
-  The Balvenie  
**39** ↑ 11%
-  The Glenlivet  
**27** ↑ 8%

**81%**

Sales: 3.2

Sales by Brand



Rum Total Sales

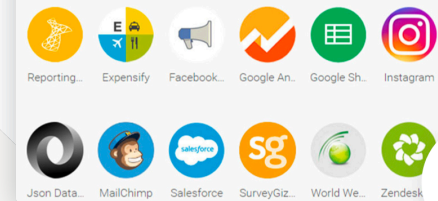
Agency	Company	Agency R.	Agency C.	Agency D.	Agency G.
1	Online	Planet Travel	Europe	Germany	Family
2	Online	Planet Travel	Europe	Germany	Family
3	Online	Planet Travel	Europe	Germany	Family
4	Online	Planet Travel	Europe	Germany	Family

### New Connection

Select your data source



Or choose a platform



# What is openness?

Technologies that allow for flexibility and provide architectural choices are referred to as **Open**. While there are different meanings and loaded interpretations of what constitutes openness, in the area of technology adoption it is firmly defined as the opposite of being closed: **Locked-in**.

Architectural choices, which make it simpler in the future for new choices to be made, can provide some protection against making the wrong decision today, as the consequences of a change of mind down the track are not as significant. A solution that accommodates this is ideal, which makes identifying and understanding what is locked-in and what is open very important.

In this whitepaper, we delve into the importance of open architectures for long-term flexibility and future-proofing when deciding to adopt a business intelligence and analytics platform. It will take you through defined phases of the overall product journey to help you action the most effective and efficient introduction of embedded analytics possible.



# What is an open architecture?

Technology architectures are in a constant state of flux. New solutions emerge, some mature and become mainstream, and others disappear. Decision-makers, meanwhile, are tasked with choosing new technology solutions which can have several long-range implications for the organizations they work for. The amount of choice available can be, quite simply, overwhelming.

1. Which technology vendors to partner with?
2. Which architectural patterns to adopt to?
3. Which platforms to standardize?

These types of major business decisions need to be made based on the knowledge we possess today, with full awareness that the basis of those decisions will change in the near future. But making the wrong decision on which technology to adopt can impact costs, customer experience, competitiveness, and ultimately the success of the organization's mission.

There is no guarantee that the right decisions will always be made, or in fact that a right decision today (based on all the available evidence) will not be the wrong decision tomorrow. Thus, flexibility and being able to rapidly respond to changes when necessary by making new choices, should always be a key part of any architectural decision to avoid being locked-in.



# Why is technology lock-in bad?

Lock-in occurs when the cost or effort of moving away from a particular choice (platform, vendor) outweighs the benefit of doing so, even if that choice is overall good for the business.

Many companies have considered or experienced that desire to take advantage of a cheaper platform, more feature rich-software, or to divorce from a vendor who is not delivering. But the pain of moving is simply too great to consider doing it. That is the state of being locked-in.

No-one wants to be locked-into a solution, unless that solution delivers compelling value on an on-going basis, with Apple being a popular example. Imagine a day in the future where Apple slips behind the pack. How hard is it going to be to unwind one's life from the Apple ecosystem?

The stakes get significantly higher when it comes to organizational technology decisions. In particular, lock-in can be problematic when choosing new software platforms. Some might argue it is in the interest of the vendor to lock you in, particularly in the age of Software-as-a-Service (SaaS) and subscription software models, where use of software requires ongoing payments. The harder it is to leave, the higher the likelihood those payments will continue into the future.

Lock-in should be distinguished from stickiness. Stickiness occurs where customers want to keep using the software, not when they are forced to. Features and usability of the platform, the commercials and the service levels are such that the customer experiences ongoing value and willingly stays with the platform while not feeling limited.

# Where can lock-in appear with software?

Lock-in can manifest in many different forms. It not only occurs when wanting to change specific software, but also when wanting to change the environment within which that software operates.

**Cloud providers** - Solutions part of a larger software stack owned by one of the major cloud computing vendors (Amazon Web Services, Microsoft Azure, Google Cloud) and only runs or runs optimally on their cloud.

**Operating systems** - Solutions typically developed for one OS platform or closed environment, such as only on Windows without support for Linux, or different versions of the software across different operating systems.

**Proprietary features and skill-sets** - Software platforms that feature proprietary scriptings and languages that essentially become throw-away skills. All software requires some degree of familiarization or training depending on its complexity and inevitably changing from a heavily proprietary solution will result in an extensive (and expensive) retraining effort.

**Software that tries to do too much** - Solutions that provide broad ranging functionality that can span beyond its initial purpose can be harder to remove and replace from the business.

# How to make more open choices in your technology adoption

First and foremost, be educated about where lock-in can occur in your organization. Ask questions throughout, be informed about the choices you make and trade-offs involved.

Consider not just changing your mind about the software you are purchasing, but also if other elements of your environment were to change. For example:

1. You move from on-premises to the cloud
2. You change cloud providers
3. You decide to move all your computing workload from Windows to Linux or vice versa
4. You change analytics database providers

Maintain a healthy level of skepticism throughout the buying process. It might be years away, but think of what the consequences of a different decision at some stage in the future might be.



# 3 examples of technology lock-in among BI platforms to avoid

Here are some specific examples of how lock-in can occur with BI platforms.

## Proprietary data storage

For many years, data cubes or in-memory database storage was a feature of business intelligence platforms. Ingesting data into the proprietary data storage layers of these tools often provided a performance boost to data discovery processes, especially where personal copies of data could be carved out from corporate data stores. This convenience, however, came at a cost: Data proliferated in an ungoverned way across organizations, but more importantly the greater the investment in these data layers, the tougher it became to switch. Data stored in these types of proprietary layers is generally only accessible from the tools provided by the BI platform provider. Organizations today are spoiled for choice with fast, analytic databases and the need to store data in proprietary formats within a BI platform is no longer necessary.

**Examples:** Tableau Hyper, Qlik Associated Engine, MicroStrategy Cubes, Sisense Elasticubes

## Cloud-aligned platform providers

The mega-cloud providers, Amazon Web Services (AWS), Google Cloud Platform (GCP) and Microsoft Azure have been in a war to win as many computing minutes from as many organizations as they can. This not only means being able to run any type of computing workload, but also by offering compelling regularly discounted software offerings that are exclusive to their platform, or work most optimally on their platform. These vendors have been developing their own offerings or gathering capability through acquisitions. The challenge comes when organizations want to move some computing workload to another provider to achieve improved price or performance; software choices they have made often lock them into a particular cloud provider, leaving them with an expensive, complex and difficult task to leave.

**Examples:** AWS QuickSight, Microsoft Power BI, Google Looker, Salesforce's Tableau, Oracle Analytics Cloud

## Proprietary skill sets

Some business intelligence platforms offer the opportunity to extend the platform's capabilities. For example, by creating custom connectors to data sources, implementing custom charts, creating custom calculations or implementing custom user interface (UI) elements. Many vendors also require developers to learn specific scripting or programming languages and frameworks. This means specific extensions will likely need to be redone if a new platform needs to be chosen, and significant time is required to learn these proprietary languages, which is completely throw-away as your team will not be able to use them once moving to new tools.

**Examples:** Looker LookML, Microsoft DAX, Qlik Expressions, ThoughtSpot Modeling Language



# How Yellowfin approaches open architectures

Yellowfin has always been built on the principle of maximizing choice for our customers. We've built out a number of capabilities to provide ample options suitable for a variety of use cases.

**Flexibility:** Because it is built on the Java platform, Yellowfin is a fully integrated and portable solution that can run on any modern operating system, run optimally on-premises, or on any cloud provider.

**Web-based:** There are no desktop components of Yellowfin; all user access is done via browser, simplifying deployment.

**Direct Query:** Yellowfin does not require the loading of an organization's data; instead, it will connect to any desired data source and run queries live against that data source.

**Extensibility:** Extending Yellowfin does not require knowledge of specific languages or proprietary scripts, as extensions can be coded using familiar developer languages like Java, JavaScript, CSS and HTML.

**Platform-agnostic:** Yellowfin's non-proprietary data preparation outputs can be used with other analytic tools, and its in-built data storytelling capability Stories can integrate dashboards and reports from competing platforms like Microsoft Power BI, Qlik and Tableau into long-form data story content.

We believe Yellowfin is one of the best examples of a business intelligence solution built around the benefits of open architecture, so much so we've been recognized for it - we recommend reading the full report of the [Gartner Magic Quadrant 2021 in Business Intelligence and Analytics Platforms](#) to see for yourself.

Ultimately, no decision can be completely future-proofed, but at Yellowfin, we want to ensure that future changes to your business can be made with minimal disruption and regret.

