The past 30 years have seen two major shifts in the prevailing technology paradigm. In the late 1980s and early 1990s, client-server applications became the accepted state-of-the-art. These combined the benefits of shared central resources – typical of the older mainframe world – with the local computing resources of increasingly powerful desktop machines. Typical applications involved a ‘fat client’ with a server database. Most of the analytical heavy lifting took place on the client, while the server provided a shared database utility. The major drawback of this approach for enterprise applications was the need to undertake co-ordinated rollout of any updates to the fat application on multiple, often far-flung, client machines.

The rise of the internet in the late 1990s ushered in the ‘thin’ client paradigm we know today. Client machines did not require specialised applications. All that was needed was a generic application capable of parsing and interpreting self-describing files on the server, displaying their content appropriately and providing facilities for users to respond as desired to centrally generated messages and queries – what we all know today as a browser.

These two approaches share one common feature, however – namely, a broad reliance on relational databases for many types of structured data storage and retrieval. Within the framework of their table definitions, these databases enabled highly complex queries to access, filter, sort, group and otherwise manipulate data in a very flexible fashion. In this sense, relational databases and structured query language represented a huge advance. They were the culmination of years of trial-and-error development of database management systems.

Relational databases continue to be a powerful and effective tool in an environment where the structure of an underlying dataset is stable. Where they fall short is in dealing with a dynamic environment where the underlying data structure itself is unstable and subject to frequent revisions. In the context of trading activities, market data conventions tend to be comparatively stable. It is primarily trade terms and conditions that evolve in response to competitive forces. Accommodating such changes requires modifications to table structures that can have knock-on effects for existing applications that process pre-existing trades. For this reason, new trade types are often not immediately and smoothly accommodated in existing trading and risk systems. This creates side pockets of special processing that often fail to be included in broader enterprise risk systems.

Data and analytical fragmentation have been risk management obstacles for decades, but efforts to overcome the problem have been rooted within the paradigm of relational databases. David Rowe argues moving beyond this framework holds the key to success.