DEEP DIVE

Data Warehouse Automation

By Stephen Swoyer

April 2019

Report excerpt prepared exclusively for:

TIMEXTENDER

Eckerson Group

Deep Dive: Data Warehouse Automation

Part I:
About Data Warehouse Automation

A Brief History of DWA

The invention of data warehouse automation (DWA) as a technology category was inevitable. And it *was* invented in a real sense: the first DWA tools evolved to address gaps in the development, deployment, and maintenance of data warehouse systems.

A data warehouse is always bespoke to a degree, custom-tailored to the size, activities, and priorities of the business. Although it does not necessarily follow that the technologies, processes, and practices underpinning and supporting the data warehouse must *also* be of bespoke design, IT practitioners nevertheless continue to design the equivalent of bespoke data warehouse systems. The practical effect is that each new data warehouse is a greenfield project, at least as regards the configuration, orchestration, and maintenance of the technologies — and to a lesser degree, of the processes and practices — that comprise it. This is the chasm that DWA as a technology category evolved to bridge.

But just as the role of the data warehouse has changed, so, too, has that of the DWA toolsets that evolved to support it. For a long time, the focus of most DWA tools was the data warehouse itself, not just because the warehouse was

the default target and *sine qua non* of DWA, but because the warehouse was the center of the business-analytic universe, at once a central repository for all business-critical information and the go-to platform for processing and productionizing analytic workloads of all kinds.

This is no longer the case. The data warehouse still occupies a vital position in the business-analytics universe, to be sure, but that position is … off to the side. The warehouse is now one among *several* potential destinations for business-critical data, one among *several* business-critical analytical platforms, and, crucially, one among several *sources* of business-critical data. Yes, almost all business-critical data will eventually find its way into the data warehouse — and the warehouse itself will continue in its role as *a* single version of the truth for users that require it — but it is no longer the focal point of data integration and analytics for most organizations.

For a long time, the focus of most DWA tools was the data warehouse itself. This is no longer the case.

DWA tools have evolved to address this transformation. Most DWA tools were designed with relational data and a relational database in mind. In practice, this bias generated an expectation of relational structures with respect to source and target systems. The specific attributes of a warehouse target might have varied[1], but data warehousing's foundational relational bias did not. The context in which data warehouse development and deployment took place was likewise predetermined: data integration for data warehouse systems was biased in favor of on-premises deployments, using SQL-compliant interfaces (ODBC, JDBC) — or any of several proprietary loading technologies — and connecting via reliable (internal) network transport. The advent of NoSQL and cloud brought radical change. The former fundamentally transformed the data integration landscape for data warehousing and analytics; the latter fundamentally transformed the site or context of data integration itself, along with the context, content, and complexity of analytics.

The two in tandem multiplied complexity, exploding the relational and on-premises biases that had been fundamental to data warehouse design and DWA.

[1] For example, over time, column-oriented RDBMSs emerged to supplement row-based RDBMSs for analytical workloads. In the same way, massively parallel processing, or MPP, databases emerged as affordable alternatives to conventional (SMP) RDBMS engines. With DWA, the focus of data integration shifted, too: most tools nominally implement an **ELT**-like process: e.g., data is extracted from source systems, moved into a landing zone (usually "on" the target warehouse), transformed, and loaded into the data warehouse. This is ELT as distinct to ETL. This distinction has to some extent collapsed, however. Today, for example, data might be extracted from one cloud context (a block storage service), moved into another (an elastic compute instance), transformed, and moved again into a staging area or landing zone in an on-premises database. At that point, the same data could undergo one or more additional transformations prior to being loaded into the warehouse.

Today's DWA tools are designed to connect to on- and off-premises data sources alike. They no longer presuppose conventional, on-premises relational sources, and they can populate relational or non-relational targets … irrespective of where they're deployed.

Today's DWA tools are designed to connect to on- and off-premises data sources alike. They no longer presuppose conventional, on-premises relational sources, and they can populate relational or non-relational targets (e.g., platforms such as Hadoop and Spark), irrespective of where they're deployed. It's probably exaggeration to say that the average DWA tool is as adept at speaking JSON as ODBC and JDBC, but today's DWA tools are polyglot to a degree that would have been unimaginable even five years ago.

There's something else, too. Like almost all cloud platforms and services, modern DWA tools are updated rapidly as their parent vendors work to accommodate the ceaseless change (e.g., the introduction of new features or API deprecations) set by Amazon Web Services, Google Cloud, Microsoft Azure, and other cloud service providers.

For this reason, the emphasis of most DWA vendors — if not the focus of DWA itself — has shifted to the cloud. Since early 2016, vendors have focused on updating or redesigning their products to better accommodate cloud sources and targets. Several vendors have even moved core pieces of their DWA products or services into the cloud.

Finally, DWA vendors have begun to appropriate at least some of the concepts, practices, and methods of DevOps, that revolution in continuous software development and delivery that has transformed the enterprise software development lifecycle. This is not to say that the average DWA tool has become a DevOps tool, or that today's DWA tools so much as pretend to achieve meaningful interoperability with DevOps tools such as Ansible, Chef, Jenkins, and Puppet. It is to recognize a consonance between DWA and DevOps as development methodologies. There's common ground — and a common purpose — that DWA vendors are just beginning to explore. The related category of DataOps offers an example of how DevOps ideas are being assimilated into analytical development. At this point, some DWA vendors are more comfortable speaking the language of DataOps than others. And no DWA tool could be considered DataOps "ready;" this last is, after all, primarily a movement of and for self-service users. Its aim is to address the needs and priorities of self-service users in a variety of roles.

Generally speaking, most DWA tools are designed with the needs and priorities of IT foremost in mind, their user experience (UX) designed for the hypothetical IT user. Nevertheless, both DataOps and DWA are coming at the same problem — the reality of continuous analytical development and delivery — from essentially opposite ends of a spectrum. It isn't a stretch to say that DataOps and DWA are destined to converge.

In any case, the market for DWA tools looks drastically different today than just three years ago. As much as things have changed, however, the logic of DWA in the on-premises world + cloud is, in crucial respects, the same as it was in the old, on-premises-only model.

Fundamentally, DWA aims to do the same things in the cloud — or in conjunction with NoSQL sources and targets — as it does in the legacy on-premises world. To understand why, let's look at what DWA actually *is* and what it is designed to do.

What DWA Is

The first data warehouse automation tools were created by consultants or integrators. In what was essentially a case of parallel evolution, different groups of people, in different places and at different times, found that they were having to do the same things over and over again as part of the nuts-and-bolts process of designing and implementing new data warehouse systems for clients. They developed their own tools to simplify and accelerate this process. As these tools grew in features and capabilities, they became more useful, and eventually the people who designed them gambled they could spin them out as successful dedicated products. Basically every extant DWA tool has an origin story like this. To this day, in fact, most large consulting or integration firms have their own in-house DWA-like tools; if they don't, they use tools from commercial DWA vendors.

A DWA tool actually does three things. First, it provides a context — a site, so to speak — in which to consolidate and organize the diverse tasks integral to the process of designing, developing for, and deploying a data warehouse. Early data warehouse development was an ad hoc, disconnected process that made use of many different tools and depended critically on human oversight, especially to manually orchestrate interoperability among all the constitutive tools. A data warehouse design project might use technologies as disparate as Microsoft Word and Excel; a data modeling tool such as erwin; one or more ETL tools[2]; and several different RDBMSs, in addition to the target data warehouse itself. This is to say nothing of the script-driven automation that was (and to a degree still is) used to tie everything together.

[2] Once these became commercially available, that is. Early ETL was for the most part a script-driven affair.

DWA vendors have begun to appropriate the concepts, practices, and methods of DevOps, that revolution in continuous software development and delivery that has transformed the enterprise software development lifecycle.

Second, DWA permits a degree of abstraction with respect to the nuts and bolts of warehouse design, development, and maintenance. A DWA tool aims to mask much of the complexity inherent in tasks as varied as the following:

- Connecting to, exploring, and extracting data from upstream sources;
- Loading source data into a warehouse staging area;
- Building or changing a logical data model;
- Generating (or re-generating) the transformations used to populate data structures in the warehouse;
- Building, changing, or maintaining the different types of denormalized data structures (star schemas, OLAP cubes) that are commonly exposed to front-end BI reporting and analytical tools;
- Determining the impact of proposed changes to upstream sources;
- Upgrading from one version of a target RDBMS to another;
- Moving from a warehouse target running in one context (e.g., on-premises Oracle) to a target running in a very different context (e.g., Azure SQL Data Warehouse).

All modern DWA tools expose ease-of-use features (in the form of user-driven GUI wizards, user-customizable automation capabilities, etc.) that are intended to accelerate these and other tasks.

Third, data warehouse automation has evolved into a **lifecycle management** solution for data warehouse systems. Conceptually, we tend to frame DWA as a technology for accelerating much of the work that accompanies the design, deployment, and maintenance of data warehouse systems. In addition to the core phases of data warehouse design and deployment (to include scoping, prototyping, testing, and ongoing development), however, modern DWA tools address a range of related lifecycle tasks, including the following:

- Maintaining a data warehouse, with a special emphasis on containing maintenance costs;
- Upgrading a warehouse or migrating from one warehouse target platform to another; and

- If necessary, retiring a data warehouse.

As a rule, DWA focuses on eliminating time-consuming, tedious, or rote tasks, especially including the creation and curation of documentation. (Most DWA tools automatically generate and manage updates for documentation and metadata.) All DWA tools generate objects or structures that are optimized for one or more DBMS target platforms; nearly all make use of a number of different optimized/DBMS-specific loading technologies.

What DWA Does

DWA vendors have begun to appropriate the concepts, practices, and methods of DevOps, that revolution in continuous software development and delivery that has transformed the enterprise software development lifecycle.

So much for what DWA is — or purports to be. What are its effects in practice? More to the point, what benefits do organizations that adopt DWA tend to realize? At a high level, the use of DWA technology is consistent with the following benefits:

- **Rapid deployment**. Acceleration — not automation — is DWA's *raison d'etre*. Absent a dedicated DWA tool, data warehouse design, development, and deployment typically involve a passel of vendor- or technology-specific tools. The work of coordinating or orchestrating the interactions among all of these tools must be performed by human beings. Put differently: *human oversight* becomes the de facto interoperability technology in traditional data warehouse development. The same is true of tasks such as managing and maintaining — i.e., *changing* — the warehouse. DWA tools expose ease-of-use features (GUI wizards; drag-and-drop gestures; automatic discovery of entities, relationships, etc.; user-customizable automation capabilities) that simplify or eliminate the need for human oversight.
- **Reuse, repeatability, and governance**. We treat code or object **reuse** as the holy grail of software development. Studies show that code reuse not only saves time, effort, and money, but that it is also associated with a range of beneficial second-order effects, including the following:
  - › superior auditability,
  - › improved governance,
  - › simplified maintenance, and

› improved software quality[3].

In practice, most of these effects have at least limited cost-reducing effects, too. In spite of its alleged benefits, studies shows that software reuse at scale is extremely difficult to realize in practice[4].

On the other hand, reuse has a somewhat different meaning in analytical development than in conventional software development. The way IT approaches reuse vis-à-vis analytical development is bound up with two (related) IT-centric priorities: **repeatability** and **governance**.

In any case, the market for DWA tools looks drastically different today than just three years ago.

When an IT person creates any analytical artifact — an ETL/ELT job; a set of data cleansing routines; one or more OLAP cubes; a data engineering workload that involves the parallel execution of a series of pipelined tasks — IT requires that this artifact (with its constitutive data, code, tasks, procedures, etc.) be instantiated as part of a process that is at once **repeatable** and **governable**. In other words, the process which instantiates the artifact must occur exactly the same way every time it is scheduled to run and, moreover, must also generate a log or record when it terminates. This is "repeatability."

If for some reason the process does not complete as expected, IT requires that it log alerts, exceptions, or other contextual information. If the process results in the movement, manipulation, and/or transformation of data, IT requires that it generate a metadata record of these changes (its "lineage") — e.g., who changed what and how — along with any additional metadata that describes any newly created objects, entities, etc. If the process is intended to be triggered on an ad hoc basis, IT requires that it hew to these same requirements, too. This is "governance." Absent a unifying DWA tool, or a completely homogeneous analytical environment, it is difficult to implement this **reuse-repeatability-governance** regimen in data warehouse and analytical development. To do so would require orchestration among several different tools, services, applications, and systems.

---

[3] See, e.g., Mohagheghi, P., & Conradi, R. (2007). Quality, productivity and economic benefits of software reuse: A review of industrial studies. *Empirical Software Engineering,* 12(5), 471-516.

[4] On one hand, open source software (OSS) development could be seen as a successful example of large-scale software reuse. See Constantinou, E., Ampatzoglou, A., Stamelos, I. (2014). Quantifying Reuse in OSS: A Large-Scale Empirical Study. *International Journal of Open Source Software and Processes*. 5(3), July 2014, 1-19. As a counterpoint, a 2005 paper by William Frakes and Kyo Kang offers an excellent assessment of the promise and peril of software reuse at scale. See Frakes, W. and Kang, K. (2005) Software Reuse Research: Status and Future. *IEEE Transactions on Software Engineering*. 31(7), July 2005, 529-536.

A DWA tool aims to mask much of the complexity inherent in scoping, designing, developing, deploying, and managing a data warehouse system.

- **Reduced costs**. Ideally, a data warehouse automation tool would pay for itself, right? DWA vendors like to think so. They claim that DWA technology reduces first- and second-order costs (via, e.g., the elimination of third-party tools and services, especially ETL tools and the expertise required to operate them) and that in some cases this savings is sufficient to justify the cost of their tools. Unfortunately, there's no empirical evidence backing this claim. However credible it might seem, the requisite quantitative research is lacking[5].

  That being said, a more likely driver for reducing costs is the increased productivity that is typically realized by the use of a DWA tool. Generally speaking, a developer, data modeler, architect, etc., will be able to do more work in less time using a DWA tool than using a mix of point solutions and custom coding. This, too, makes sense. In addition to automating certain rote or repetitive tasks, DWA tools expose ease-of-use features (e.g., user-driven wizards) that can potentially accelerate warehouse design, prototyping, development, deployment, and maintenance.

  One other important consideration is **technical debt**. It's a key contributor to IT costs, particularly in connection with IT systems maintenance. A DWA tool cannot eliminate technical debt — all technologies contribute to the accumulation of this debt — but it *can* help to limit its effects.

  DWA technology is intended to promote **standardization**, **reuse**, and **repeatability** in the context of data warehouse design, development, management, and maintenance. To this end, a DWA tool aims to eliminate (or, at least, to minimize) the need for human-maintained artifacts, including scripts, procedural code, macros, documentation, or virtually any other supporting or enabling asset that requires human oversight or ongoing maintenance. In typical data warehouse development, all of these artifacts are potential sources of technical debt. Generally, they're specific to diverse tools, applications, and middleware; some, such as shell scripts, are even specific to operating system-level services, such as cron. Absent a DWA tool, IT must maintain all of these assets manually. Collectively, this amounts to a significant interest payment on ballooning technical debt. Think of DWA as a strategy for refinancing this debt.

[5] One of the few available peer-reviewed studies specifically addresses the use of *automated ETL tools* in data warehouse development. See Rahman, N. & Rutz, D. (2015). Building Data Warehouses Using Automation. *International Journal of Intelligent Information Technologies*. 11(2), April-June 2015, 1-22.

Data warehouse automation has evolved into a lifecycle management solution for data warehouse systems.

For the most part, an organization that adopts a DWA tool should expect to realize all of these benefits to some degree. In a real sense, however, we're still treating primarily in high-level abstractions. The way to get a feel for what data warehouse automation technology can actually do is to put one DWA tool through its paces — or even several.

In the sections that follow, we'll put biGENiUS's DWA tool to the test. How does DWA technology promise to simplify and accelerate the process of building the data warehouse? More important, does DWA technology make it easier and faster to custom-fit the data warehouse? How does DWA simplify the process of maintaining (e.g., making changes to) a production data warehouse? Let's get down to the nitty gritty of how DWA works in practice.

Part II:
DWA Tools in Context

TimeXtender Discovery Hub

To read additional product profiles, see the full report.

TimeXtender was founded in 2006, with headquarters in both Denmark (Aarhus) and the United States (Seattle). It is unique among data warehouse automation vendors in that it focuses exclusively on Microsoft's business intelligence (BI) and data warehousing stack.

Half a decade ago, this primarily entailed a focus on SQL Server and Office (now PowerBI) BI stacks, with support for several versions of SQL Server as de facto data warehouse targets. With the advent of Microsoft's Azure cloud platform, however, TimeXtender's remit expanded significantly, such that it now supports Azure SQL Database, and Azure Data Lake as target platforms, as well as connectivity to and interoperability with a number of related Azure BI or data integration services.

Discovery Hub platform accelerates… and automates… most aspects of the design, development, deployment, management, and maintenance of a warehouse running on Microsoft's SQL Server and Azure SQL platforms.

TimeXtender says its Discovery Hub platform accelerates — and, to the degree practicable, automates — many if not most aspects of the design, development, deployment, management, and maintenance of a warehouse running on Microsoft's SQL Server and Azure SQL platforms. Discovery Hub likewise promises to simplify upgrades from one version of one platform to another (e.g., SQL 2008 to SQL 2017); from one platform context (SQL 2017) to another (Azure SQL); or — as of SQL Server 2017 and beyond — from SQL Server running on Windows Server to SQL Server running on a supported Linux distribution.

The Discovery Hub Architecture

TimeXtender positions **Discovery Hub** as a database abstraction layer. It comprises three logical components: an **Operational Data Exchange** (ODX), a **Modern Data Warehouse** (MDW), and a **Semantic Layer**. In the Discovery Hub architecture, each of these components is optional; an organization can opt to build a conventional data warehouse or conventional data marts. According to TimeXtender, the logical components of the Discovery Hub reference architecture provide the foundation for a combined data and analytics platform. In this scheme, the ODX functions as a central repository — a kind of all-purpose data ingestion tier — to take in and store data of virtually any type.

In the Discovery Hub reference architecture, the ODX serves as an Ur source for the **MDW**, analogous to a presentation layer. At a high level, an organization builds the MDW by modeling and conforming the raw data that it ingests in the ODX and using this data to build data models optimized for analytics consumption[1].TimeXtender describes its MDW data model as a richer, more granular take on the conventional star schema; for example, the MDW model incorporates data from additional tables (e.g., transaction header tables and reference tables), which TimeXtender says enhances its support for the self-service use case.

…the MDW model is ideal for most organizations.

That being said, a modeler can use Discovery Hub to generate virtually any type of model, from a conventional star schema dimensional model to something analogous to a Data Vault model[2]. TimeXtender maintains that the MDW model is ideal for most organizations. It notes that the technological constraints which once hindered the use of a snowflake-like schema — which is similar to its MDW model — have largely been addressed, thanks (for example) to commodity in-memory and parallel processing technologies.

Practically speaking, the Discovery Hub MDW consists of both a data presentation layer and a **data staging area** (**DSA**) in which data is first modeled and conformed. Again, creating an ODX and building an MDW is not a prerequisite for using Discovery Hub; an organization can also use the product to support development of a 3NF warehouse, a Kimball-esque warehouse, or something else entirely. In place of TimeXtender's reference MDW architecture, developers would use Discovery Hub to create a staging area, design an appropriate data model, generate ELT scripts, and build a presentation layer or subject-specific data marts.

Like everything else in Discovery Hub, from the instantiation of the ODX to the creation of the MDW, TimeXtender consolidates all of this into a point-and-click, wizard-driven process.

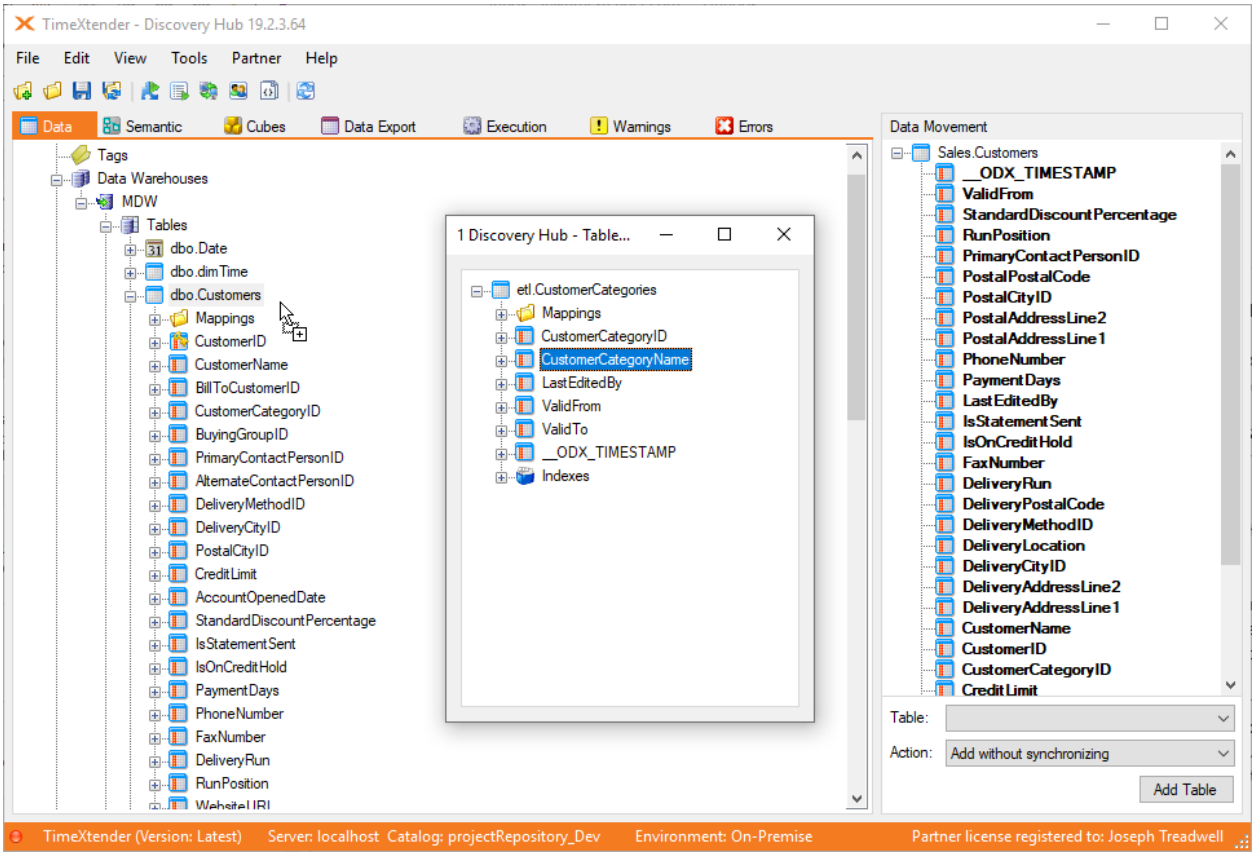
---

[1] TimeXtender recommends the creation of partially denormalized physical models for maximum compatibility with Qlik Sense, Microsoft PowerBI, Tableau, and other self-service BI tools.

[2] Data Vault modeling is quite unique and implements structures (e.g., links, hubs, and satellites) that are not used in other types of models. Moreover, Dan Linstedt, its creator, is justifiably proprietary in his insistence that software vendors certify their products and services as compliant with Data Vault 1.0/2.0 concepts and methods. (A scheme that purports to implement a Data Vault-like structure but which is plagued by issues in practice reflects poorly on Linstedt's creation. This is one reason he emphasizes the importance of certification.) In other words, Linstedt strongly discourages the use of any tool/product that is not certified as Data Vault-compliant.

Building the Data Warehouse

To get a sense for Discovery Hub's bona fides as a data warehouse automation tool, let's walk through creating a data warehouse from scratch. The first step is to instantiate an ETL tier or ODX on a supported platform: Azure Data Lake for the purposes of this walk-through. In Discovery Hub, this is a fairly straightforward process: i.e., a matter of creating and configuring a new object called "ODX" and deploying it in Azure Data Lake. Discovery Hub's user interface (UI) and user experience (UX) are similar to those of Microsoft's Active Directory: a user clicks on an object in a tree-view in the left-hand pane and makes changes to it in the right-hand pane. Drag-and-drop operations are supported between objects; the user right-clicks on an object to expose advanced options.



Once she has created an ODX, the user invokes another GUI wizard to configure access to upstream data sources, selecting (if applicable) from among the connectivity "providers" that TimeXtender bundles with Discovery Hub. To its credit, TimeXtender offers dozens of free providers with Discovery Hub; in addition to supporting access to common on-premises sources (DB2, Oracle, SAP), it bundles providers for Google Analytics and Salesforce.com, among

other sources. In addition, Discovery Hub integrates more than 90 ADO.net providers from CData Software; all of these adapters can be downloaded for free, installed, and configured from within Discovery Hub itself. Similarly, Discovery Hub supports other third-party ADO.net, ODBC, and OLEDB adapters. Finally, Discovery Hub exposes an ADO.net connectivity wizard (called "Any Source" in TimeXtender's lexicon) that can facilitate access to proprietary or otherwise unsupported data sources.

Once a user has built and deployed an ODX, she can use it to feed her MDW. In the Discover Hub reference architecture, this is actually a two-step process, entailing as it does the creation of a discrete MDW database — basically a data presentation layer that can be accessed by BI reporting and analytical tools — along with that of a(n optionally) separate DSA database in which data can be modeled and transformed prior to instantiating it in the MDW.

This, too, is consolidated into a wizard-driven task: a developer fires up the Discovery Hub GUI tool, right-clicks on the appropriate object, selects "Add Data Warehouse," and repeats this process to create MDW and DSA databases. Once this is done, the task of defining dimensions (Product, Customer, Date) and creating a fact table for the MDW is just as straightforward. Discovery Hub exposes all data sources as nested objects under the ODX; a developer can (e.g.) find an individual database table in the ODX and drag and drop it into the DSA. In the DSA context, she can manually select each of the tables, dimensions, and attributes to bring forward. (Because Discovery Hub creates mappings for each table, the user can modify a table's structure without changing its *actual* structure in the DSA.) A user can drag and drop fields from one table to another in order to (e.g.) indicate relations or create conditional lookups. At any point in this process, Discovery Hub can automatically (re)discover and (re)generate relationships, too.

Once she's imported her dimensions, it takes just a few clicks to instantiate them in the MDW. Discovery Hub automatically validates the MDW model, but not the data itself. Instead, a developer uses Discovery Hub's "Query Tool" wizard to write SQL queries that validate data[3].

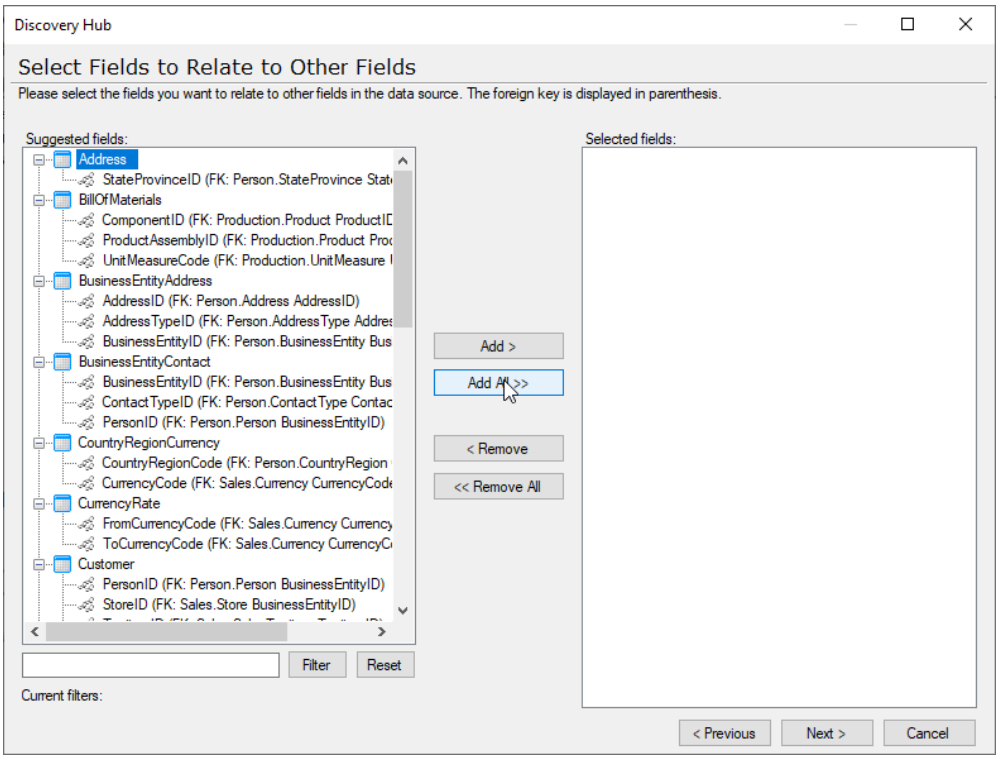
The user repeats this process to populate the product, customer, date, etc., dimensions of the MDW model. She uses the same basic concepts and methods to build the **Semantic Layer**, which consists of subject-specific models (populated with relevant business terms and definitions) that can be pushed

[3] Developers are accustomed to coding their own data validation checks. A developer can also use TimeXtender's built-in data quality rules to enforce checks for uniqueness, referential integrity, counts, etc.

out to self-service BI tools. Using Discovery Hub, a user can create a single semantic model and push optimized versions of that model out to several different front-end tools. Discovery Hub's GUI helps accelerate much of this work, and its automation capabilities likewise accelerate other core tasks (e.g., identifying relations; building, validating and instantiating data models; creating documentation, etc.)

Wizard-driven acceleration and automation capabilities can accomplish just so much, however.

Another important dimension of a DWA tool is the extent to which it simplifies and accelerates the work of custom-fitting the data warehouse. In practice, this entails exposing features to simplify tasks such as advanced data modeling, advanced data cleansing and data transformations, troubleshooting, and simplifying maintenance.

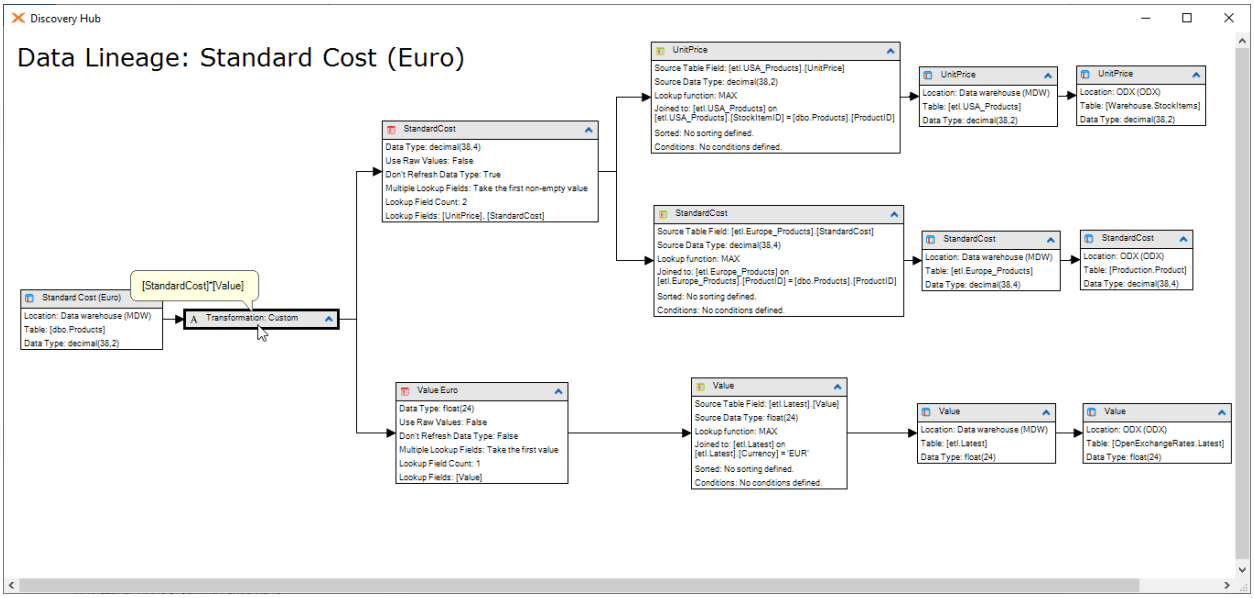


Custom-Fitting the Data Warehouse

In TimeXtender, as in other DWA tools, it is possible to rapidly design, build, test, and deploy a basic data warehouse. No organization actually *uses* a basic data warehouse, however; data warehouse design is always to some extent a bespoke proposition, and no DWA tool can completely eliminate this "bespokeness." At best, a DWA tool can meaningfully reduce the manual work required to get a

data warehouse up and running. To this end, Discovery Hub provides an array of features designed to accelerate critical areas of data warehouse design, development, and maintenance.

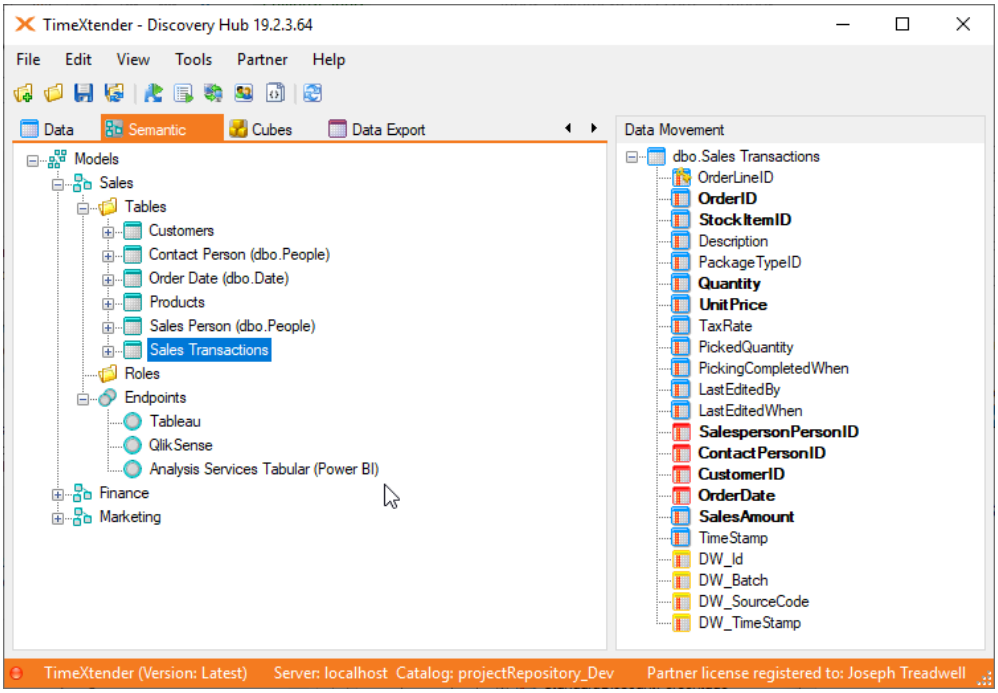
When a developer configures a relational data source mapping, for example, Discovery Hub automatically identifies the relationships between its constituent tables. As she's building the dimensions in her MDW model, however, a developer might discover that the relationships between one or more nominally related fields are not actually instantiated (as relations) in the model. She can invoke a combination of Discovery Hub's built-in tools to correct this problem. To cite one example, Discovery Hub provides a visualization tool that depicts the relationships between tables. This and other visualizations can help developers troubleshoot problems.



The emphasis isn't necessarily on automating data warehouse development — again, automating the custom-fitting of a data warehouse has limits — but on simplifying time-consuming or repetitive tasks to help accelerate that custom-fitting. This is true, for example, of the critical work of building and deploying a semantic model for use with different BI front-end tools. Once a developer builds her semantic model, she can use Discovery Hub to automatically deploy it to several different BI front-end tools, or "endpoints." By default, Discovery Hub generates Tableau and Qlik models, as well as multidimensional and tabular models that can be consumed by Microsoft's SQL Server Analysis Services (SSAS). It also exposes a template that a developer can use to build custom endpoints

which support other BI tools and services, too. In this way, Discovery Hub simplifies the task of building, testing, validating, and deploying dimensional and semantic models.

Similarly, Discovery Hub helps simplify critical and potentially time-consuming tasks, such as creating custom data transformations or data cleansing routines. Discovery Hub exposes GUI features that enable developers to apply common transformations (Upper, Lower, TrimLeft, TrimRight, ReverseSign, TimeOnly, DataOnly, Replace, etc.) to data. Users can also combine transformations with conditions (Empty, NotEmpty, IsNumeric, IsNotNumeric, Equal, NotEqual, etc.), as well as define both **custom transformations** and **custom conditions**.



Other GUI shortcut features help to simplify other essential tasks, such as creating surrogate keys, defining and managing business rules, configuring slowly changing dimensions, creating and instantiating stored procedures, etc. Discovery Hub can't automate these tasks; no DWA tool can. Instead, it provides user-configurable shortcuts designed to help simplify them.

This is true of the process of maintaining the data warehouse, too. To add or change dimensions in the MDW data model, for example, a user must repeat some of the steps outlined earlier. This is basically a process of making the desired changes and then having Discovery Hub generate a new data model, ETL, and warehouse data structures. This process is accelerated rather than

automated; meanwhile, certain time-consuming or especially complex tasks (generating ETL mappings that comport with the data model) are automated, as is the updating, deletion, or creation of metadata and documentation.

On these terms, Discovery Hub is as good as advertised. A comparison of its features and overall user experience with those of Microsoft's own BI and data warehousing stack provides a helpful contrast. In theory, Microsoft's stack is least likely to benefit from a DWA tool: in the past, after all, the company owned and controlled every layer of its combined stack, namely, the operating system, application, middleware, and back end. Surely it could deliver the equivalent of DWA-like simplification, acceleration, and common-sense automation on its own stack? Ironically, this was never the case with respect to Microsoft's on-premises SQL Server database, even with the introduction of Microsoft's otherwise formidable SQL Server Management Studio (SSMS). At a minimum, a user would also want to install SQL Server Data Tools (SSDT) to simplify the creation of packages for SQL Server Integration Services, SQL Server Analysis Services, and SQL Server Reporting Services. In practice, she would have to switch between SSDT — which runs in the context of Visual Studio — and SSMS to perform all of the tasks Discovery Hub consolidates into a single tool. She'd start by using SSDT to create SSIS packages to extract data from upstream sources and land it in a staging area she'd created in SSMS. From there, she'd use SSMS's "Install Diagram Support" wizard to kick off the protracted process of building her logical data model — beginning with the manual definition of data types. After using SSMS to manually instantiate her dimensions and fact table, she would next manually specify the relationships between her dimensions and the fact table. Eventually, she'd get to the point where she could manipulate a logical model in SSMS.

Discovery Hub doesn't just simplify, accelerate… and automate… much of the work of designing, building, custom-fitting, and maintaining the data warehouse, it also provides a central context…

There's a lot more manual grunt work involved in this and other phases, of course. A DWA tool such as Discovery Hub doesn't just simplify, accelerate, and automate much of the work of designing, building, custom-fitting, and maintaining the data warehouse, it also provides a central context — a single tool — in which to do this. The earlier walk-through outlined the process of using Microsoft's native tools to design, build, and custom-fit a data warehouse in an on-premises SQL Server environment; doing so in the Azure cloud entails the use of a quite different set of tools, however. In one sense, it's notionally easier to build and custom-fit a data warehouse in Azure SQL Database or SQL Data Warehouse; in another it's more complex, especially with respect to scaling the MPP-based SQL Data Warehouse.

Practically speaking, an organization that intended to pursue a hybrid cloud/on-premises strategy would be forced to bifurcate development between at least two different teams, each with different skill sets, tools, and priorities.

In Discovery Hub, TimeXtender offers a single tool that permits a coordinated development effort that spans both the public cloud and on-premises deployments.

Final Thoughts

The era of the DWA "tool" might actually be over. To wit: TimeXtender positions Discovery Hub as an "integrated data management platform" — a description that exceeds the remit of a "mere" data warehouse automation tool with respect to its role, scope, and purpose.

The Discovery Hub architecture is at once forward-thinking and common-sense.

Conceptually, this makes sense: Discovery Hub is intended to function as a foundation for analytical use cases of every kind, from traditional data warehouse development to self-service BI, discovery analytics, and data-scientific research and development. The Discovery Hub architecture is at once forward-thinking and common-sense. It's so common-sense, in fact, that the rest of the industry seems to be coming around to its logic, if not its inevitability.

Discovery Hub seems built for the future — or the not-so-distant future — and not necessarily for the present. For example, TimeXtender stresses that the ODX is an optional piece of the Discovery Hub reference architecture. Essentially, it's a kind of managed data lake that gives an organization a means to ingest and persist data without altering it in any way. So long as the ODX lives in SQL Server or in its Azure SQL family, an organization is relatively limited with respect to the types of data it may usefully (or cost-effectively) store in the ODX.

By shifting the site of the ODX to an elastic context such as Azure Data Lake, however, its potential usefulness and applicability grow significantly, especially as an Ur source of data for business analysts, statisticians, data scientists, and other self-service users.

The logic here is simple enough: almost all classes of advanced analytics — from basic BI discovery to data mining to the highly iterative training of machine learning models to the complex work of AI research and development — require raw data. Even the normalized data in the Inmon-esque 3NF data warehouse is too conformed — too stripped of detail — to suit the requirements of most types of advanced analytical development.

The raw transaction data that an organization persists in the ODX can be made available to other information consumers as appropriate. It can also be combined with strictly structured or polystructured data ingested from other sources.

Situating an ODX in Azure Data Lake gives self-service users, especially, a single context in which to discover and prepare data and likewise makes it easier to access relevant data. It also can minimize the amount of data that is created, moved, and persisted as part of self-service discovery, data ops, and other practices. For example, self-service users can prepare their own (derived) data extracts in the ODX and, if necessary, move them to another platform context or storage layer; they can also build pipelines to make them consumable by other applications, services, or in-memory compute engines. The ODX generates metadata, keeps track of lineage, and enforces governance.

It's still early days, too. TimeXtender introduced support for Azure Data Lake as an ODX source just last year. And Discovery Hub is still essentially an IT-oriented environment: its UX is designed with the IT persona in mind; its features and capabilities are primarily geared to the needs and priorities of IT, *not* those of self-service BI or data ops practitioners.

The upshot, then, is that Discovery Hub aims to promote rapid development, to simplify repetitive tasks, and, more important, to address IT-centric priorities such as reuse, repeatability, security, and governance. In this way, it is very much an IT-oriented tool. This is not intended as a knock. It's a recognition of the fact that — as much as TimeXtender's vision for Discovery Hub has expanded — the company, like its competitors, still has work to do. In the near term, TimeXtender expects to introduce a self-service portal for Discovery Hub that will permit self-service users to discover and prepare their own data extracts. In the longer term, it will probably broaden its development effort to focus on both IT and self-service users.

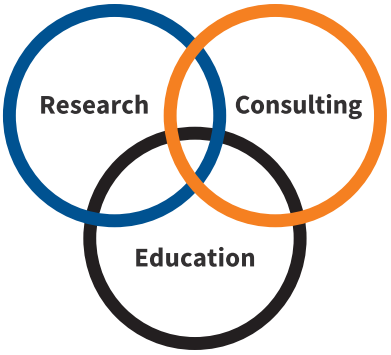
About the Author

Stephen Swoyer

Senior Analyst at Eckerson Group

Stephen Swoyer is a researcher and analyst with more than 20 years of industry experience. His research has focused on business intelligence (BI), data warehousing, and analytics, along with the larger data management segment, for 15 years. Swoyer's work combines a deep interest in BI and analytic technology with an alertness to the needs and priorities of the people who must use (or, often as not, dis-use) it. He's a recovering philosopher and occasionally guest lectures on Kierkegaard, Nietzsche, and Heidegger at both Penn State and Vanderbilt.

About Eckerson Group



We Help Analytics Leaders Succeed

Eckerson Group has three main divisions:

- **Eckerson Research** publishes insights so you and your team can stay abreast of the latest tools, techniques, and technologies in the field.
- **Eckerson Consulting** provides strategy, design, and implementation assistance to meet your organization's current and future needs.
- **Eckerson Education** keeps your data analytics team current on the latest developments in the field through three- and six-hour workshops and public seminars.

Unlike other firms, Eckerson Group focuses solely on data analytics. Our veteran practitioners each have more than 25 years of experience in the field. They specialize in every facet of data analytics—from data architecture and data governance to business intelligence and artificial intelligence. Their primary mission is to share their hard-won lessons with you.

Our clients say we are hard-working, insightful, and humble. We take the compliment! It all stems from our love of data and desire to serve—we see ourselves as a family of continuous learners, interpreting the world of data for you and others.

Accelerate your data journey. Put an expert on your side.
Learn what Eckerson Group can do for you!

Contact Us

Schedule a Call

About TimeXtender

TimeXtender — and our integrated data management platform, Discovery Hub® — empowers customers with instant access to data, enabling them to make quality business decisions with data, mind and heart. We do this for one simple reason: because time matters. A Microsoft Gold Certified Partner, TimeXtender serves its 3,000+ customers, from mid-sized companies to Fortune 500, through its global network of partners. TimeXtender was founded in 2006 and is privately owned, with headquarters in Denmark and the U.S. and regional offices around the world.

Discovery Hub is a high-performance data management platform, anchored on automation, that enables digital transformation. Discover Hub accelerates your time to data insights by up to 10 times - enabling your transformation to a data-driven business. Discovery Hub allows you to connect to various data silos, catalog, model, move, and report on the full lifecycle of data — in a single application that supports core analytics, the modern data warehouse, IoT, AI and more.

Developed with a cloud-first mindset, Discovery Hub provides a cohesive data fabric across Microsoft on-premise technology and Azure Data Services, eliminating the need to manually stitch together a patchwork of tools for data access, modeling and compliance. This creates a future-proof modern data estate, that naturally evolves with technology advancements, allowing for continuous improvement of data insights for business decision makers.

Learn more