DEEP DIVE

Data Warehouse Automation

By Stephen Swoyer

April 2019

Report excerpt prepared exclusively for:

biGENiUS®

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

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

to say nothing of the script-driven automation that was (and to a degree still is) used to tie everything together.

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).

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.

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.

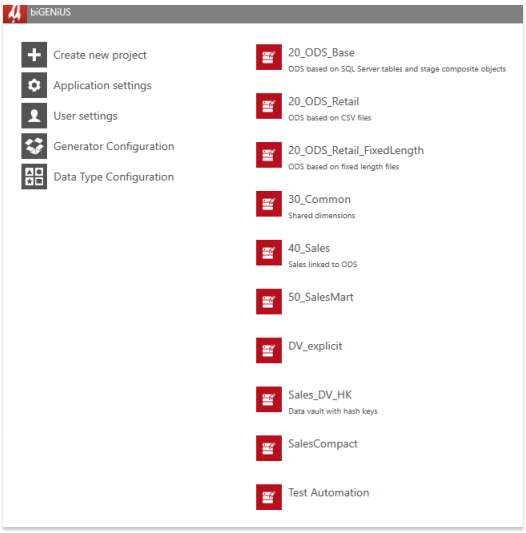
Part II:
DWA Tools in Context

biGENiUS

To read additional product profiles, see the full report.

biGENiUS is a relative newcomer to data warehouse automation. Its roots go back to 2005, when Swiss consulting and services specialist Trivadis began promoting its homegrown DWA technology as a differentiator with clients. In 2011, Trivadis launched biGENiUS as the technology's commercial incarnation. The company marketed biGENiUS exclusively in the European Union for a few years prior to expanding. Last year, it hired Gertjan Vlug, co-founder of DWA pioneer BIReady (which was acquired by Attunity in 2014) as director of sales and business development for international markets. biGENiUS launched in the United States in October.

The overall combination makes for a clean and intuitive development environment that allows for significant customization.



The product's front-end GUI resembles a cross between a modern integrated development environment (IDE) and an extensible source-code editor such as Atom, which is especially popular among open source developers. The GUI features of the front-end tool expose wizards that simplify and, to the degree practicable, automate key aspects of data warehouse design, development, deployment, and maintenance. The built-in source-code editor enables a developer to directly modify or enhance the SQL code that is generated by biGENiUS's GUI wizards. The overall combination makes for a clean and intuitive development environment that allows for significant customization.

Like all DWA tools, biGENiUS is a metadata-driven product. It stores all content associated with a data warehouse development project in its own metadata repository. This includes anything a developer creates in the process of designing and developing the data warehouse: data relating to artifacts such as upstream data mappings, data models, ETL mappings and transformations, business rules, and stored procedures.

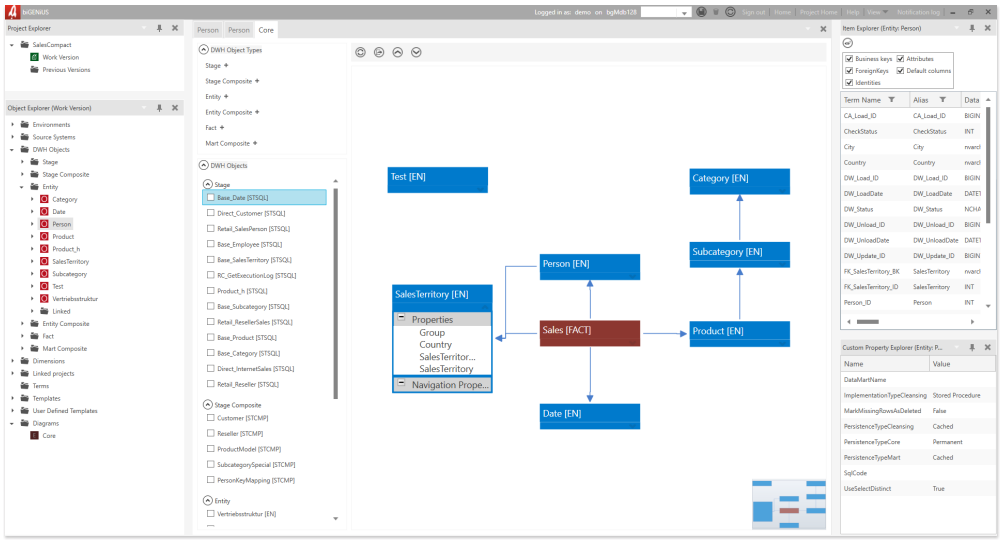
In biGENiUS, for example, a developer can make any necessary or requested changes … and regenerate the data warehouse running one or more scripts.

Metadata-driven design is so common among DWA tools as to constitute a de facto best practice. Its sheer popularity seems to support the claim that a metadata-driven approach helps to simplify the development, customization, and maintenance of the data warehouse. In biGENiUS, for example, a developer can make any necessary or requested changes — e.g., create a new surrogate key; add or modify dimensions; manage slowly changing dimensions — and regenerate the data warehouse running one or more scripts. Yes, it can take considerable time to repopulate even a small data warehouse. But this is a quantum improvement over the task of redesigning, rebuilding, and *then* repopulating the data warehouse from scratch using traditional tools and methods.

BoBYOM: Bring or Build Your Own Model

The biGENiUS front-end GUI provides a single environment in which to manage all phases of data warehouse design and development. The GUI tool runs on one or more Windows systems; other desktop environments are not yet supported. The guts of biGENiUS is its metadata repository, which stores both the data model itself and the configuration data that is relevant to the data model's technical implementation. biGENiUS says this approach permits a developer to easily transform a data model to suit a range of configurations.

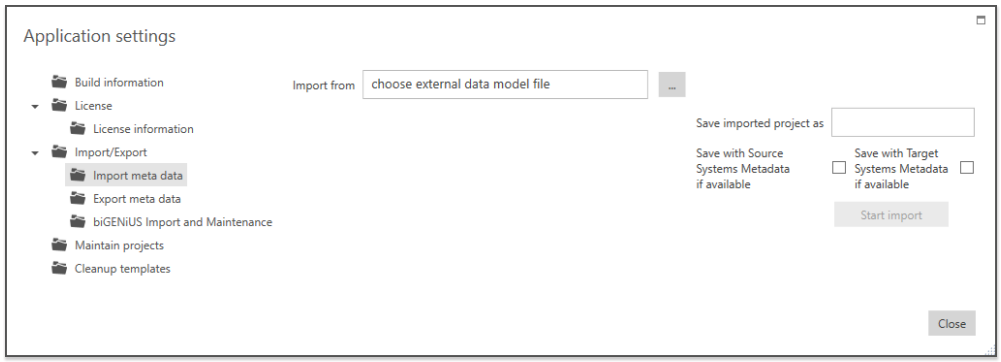
Most DWA tools claim to support agile methods to some degree. Some vendors champion agile software development or project management. Others treat agile as one of several supported schemes for software development. biGENiUS is in the second camp, although the company claims its GUI front end supports the collaborative user experience that characterizes agile methodologies.



DWA tools also tend to split on the issue of data modeling. Some are designed to consume the output of an enterprise modeling tool, such as PowerDesigner or erwin. In this scheme, the DWA tool takes a logical model that has been generated, for example, by erwin, and translates it into a physical model. The developer makes any necessary changes to this model and the DWA tool uses it to build and populate the data warehouse. This approach is consistent with a **top-down** data modeling scheme[1]. Other DWA tools emphasize a **bottom-up** approach. The idea is that IT people and, sometimes, business people will use the tool to explore upstream systems and build a model based on what they find. This process is analogous to reverse-engineering the business and its operations.

biGENiUS accommodates both business- and data-driven modeling on more or less equal footing.

biGENiUS tolerates either approach. A developer or modeler can import a logical data model from erwin and use it as a source for building a model in the biGENiUS environment. Customization of this model is accomplished through drag-and-drop and point-and-click GUI gestures. But the biGENiUS GUI likewise exposes wizards and visualizations that developers, modelers, architects, and even business people can use to explore sources as part of the bottom-up process of reverse-engineering a data model. The upshot is that biGENiUS accommodates both business- and data-driven modeling on more or less equal footing.



Building the Data Warehouse

The work of building the data warehouse with this biGENiUS is straightforward enough.

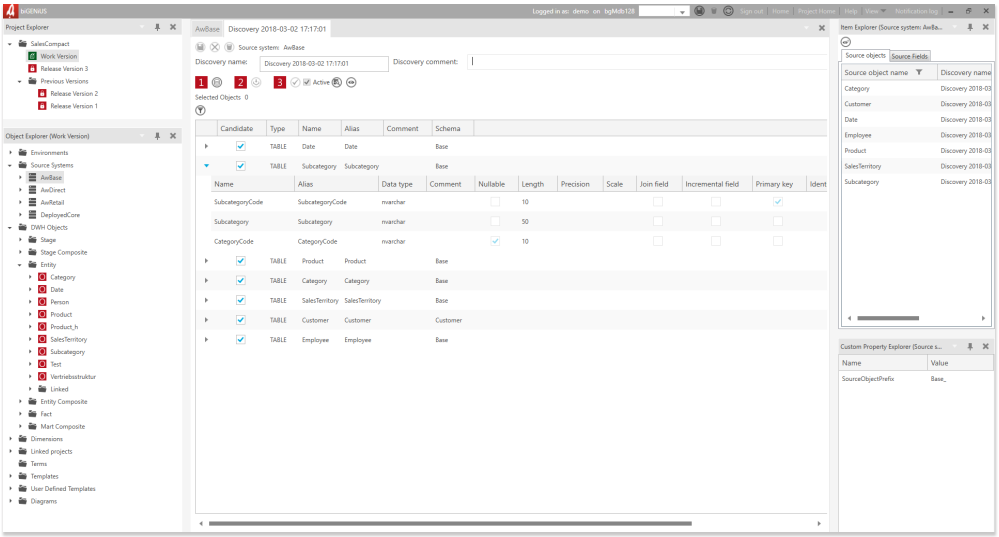
biGENiUS's home screen displays a list of all projects in its metadata repository. Clicking on an existing project opens it; clicking "Create new project" permits a user to start from scratch. Once in a project, biGENiUS exposes a "Project Explorer" view in its left-hand pane. Lower in the left-hand pane is the "Object

[1] In the same way, a modeler or architect can use biGENiUS to import an industry-specific or use-case-specific reference model (sometimes called a "blueprint"). After importing a reference model, the modeler or architect would use biGENiUS's built-in modeling tools to customize it to the organization.

Explorer." biGENiUS supports robust versioning capabilities — what software developers refer to as "version control" — and the Project Explorer pane contains information about all the available versions of a project. A user-defined "Work version" is presented by default, and the user clicks "Previous Versions" to access other iterations of the project. The Object Explorer displays a view of all objects associated with the current project. By default, it opens to the Work version.

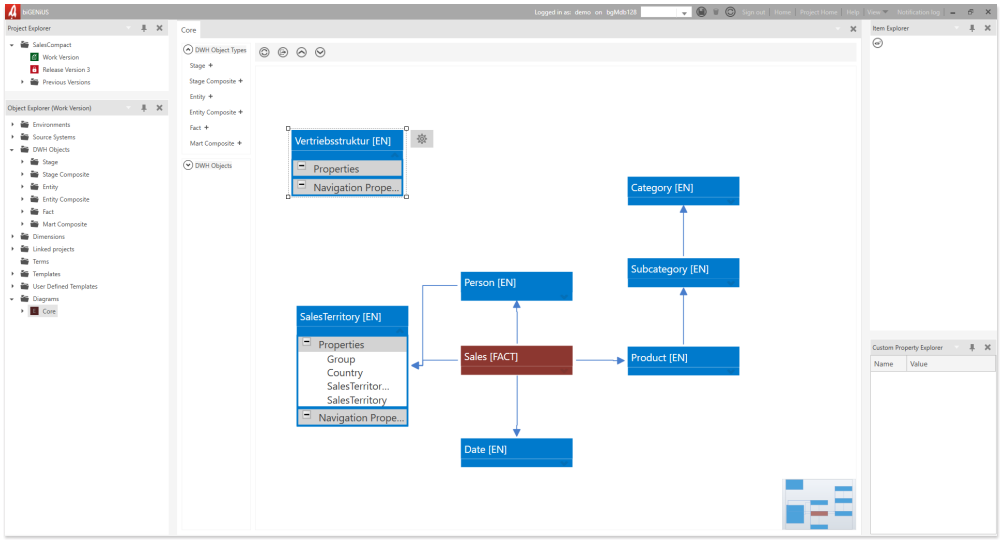
The Object Explorer is the epicenter of development in biGENiUS. Say, for example, a developer wants to provision a data source to kick-start data warehouse development. In biGENiUS, as with most DWA tools, this is a point-and-click operation: The developer clicks "Source Systems" in Object Explorer and uses a GUI wizard to create a data source object, then enters the connection string, access credentials, etc., in the appropriate fields.

biGENiUS then goes to work, automatically scanning and profiling the data source and generating a snapshot of its structures: tables, views, attributes, etc. The user clicks on the "Discoveries" tab to see all available snapshots of the source data structures, as well as to drill down into the underlying metadata. If something about the structure of a source changes, she can have biGENiUS rescan it to create a new discovery snapshot.



biGENiUS offers a couple of built-in tools for designing the data model. In business-driven (i.e., top-down) modeling, the modeler might begin by clicking "DWH Objects" (one of a series of nested objects in the Object Explorer pane) to create her entities, facts, and relationships. However, she can also use the "Diagrams" facility. Diagrams is a visual modeling tool (similar in function to a data modeling tool) in which a modeler uses right-click and click-and-drag gestures to create new entities, facts, and relationships. In the Diagrams

environment, as in a tool such as , the modeler can literally “draw” relationships between entities, between entities and facts, or among other modeling objects such as hubs, satellites, and links.



In data-driven (bottom-up) modeling, the modeler must first create and populate the equivalent of a warehouse staging area. She does this by generating “Stage Objects,” a fairly straightforward task. The modeler clicks on a data source object, scrolls through and selects from a list of pre-populated “source objects” — basically, source objects such as Product, Customer, and Date — and right-clicks to create a Stage Object. Typically, biGENiUS extracts all relevant source objects as part of the process of creating a snapshot during its discovery of upstream data structures. It likewise populates the new Stage Object with the appropriate column names, which it also extracts during discovery. In addition, biGENiUS generates a list of “technical columns” (analogous to “Data Movement Columns” in the erwin environment, for example) along with a list of source field mappings, all of which can be modified via GUI.

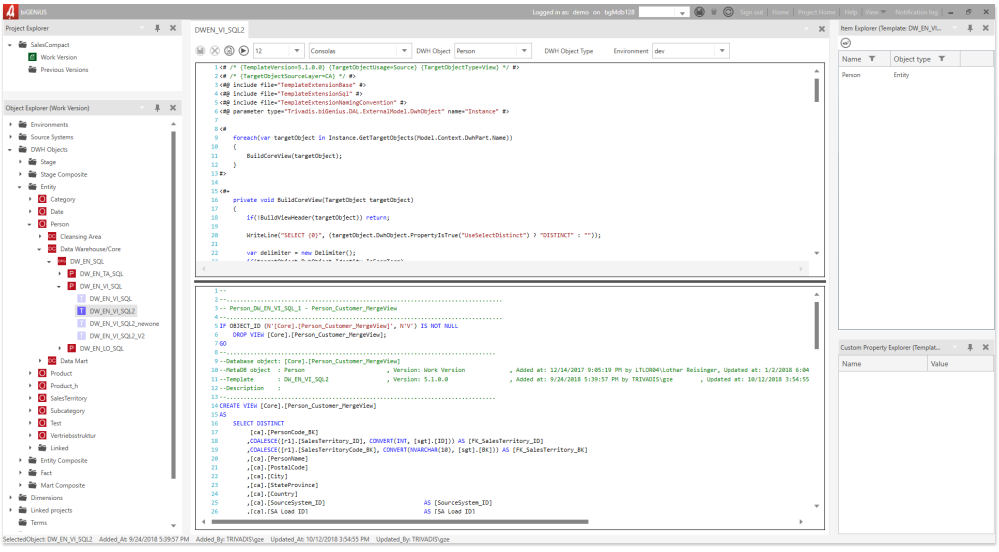
Now the modeler gets down to the nitty gritty of building her model. This is a process of right-clicking on Stage Objects and using a wizard to generate DWH Objects. She can create either an “entity” or a “fact” DWH Object, which are analogous to dimension and fact tables in a physical data model. From there, she can use the Diagrams tool to define the relationships between these objects. Once she populates her data-driven model with entities, facts, and relationships, the modeler uses a wizard to create the core data flows that map data Stage Objects (i.e., data in the warehouse staging area) to the entities in her model.

biGENiUS makes data-driven modeling pretty painless.

Putting aside its unfamiliar terminology, biGENiUS makes data-driven modeling pretty painless. Under the hood, a modeler simply configures biGENiUS to extract data from one or more upstream sources and load it into a staging area. From there, she explores the source tables, columns, and fields; creates the equivalent of dimensions and facts; and then specifies the relationships.

biGENiUS doesn't force the modeler to choose between either business- or data-driven modeling. She could easily import a logical model (or reference blueprint) and then switch to a data-driven modeling approach to make customizations. The primary difference between the two approaches is that, in business-driven modeling, biGENiUS uses a logical model (or reference blueprint) as a source to generate the relevant dimension and fact tables; in a data-driven approach, the modeler first creates Stage Objects — i.e., snapshots of upstream data source structures — and then uses these to reverse-engineer her Product, Customer, Date, etc., dimension and fact tables.

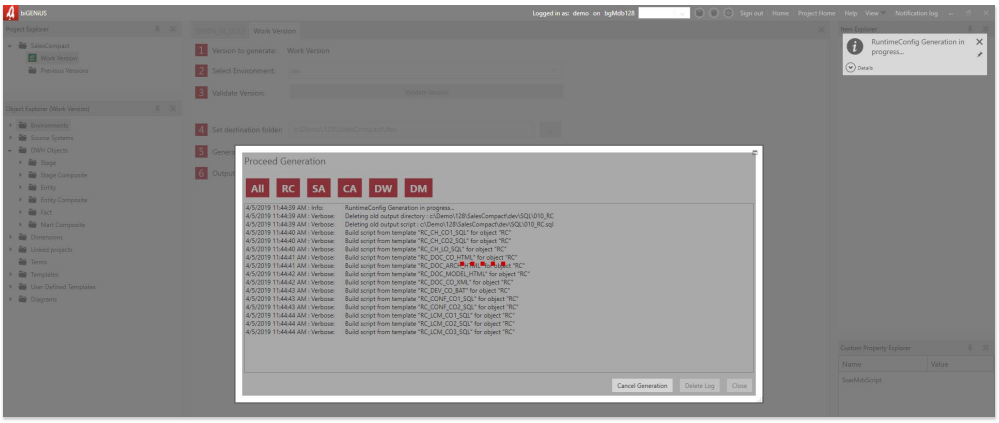
In other words, once a data-driven modeler maps a stage object to an entity, she uses the same tools and techniques as would a business-driven modeler. In both cases, the modeler customizes entities, facts, and relationships by right-clicking objects in the Object Explorer pane or by using biGENiUS's visual Diagrams environment. The modeler can specify additional attributes, define custom data types, add custom business rules and/or business logic, configure Type 1 or Type 2 slowly changing dimensions, and make a number of other changes.



Creating and deploying the data warehouse is likewise straightforward. In biGENiUS's Project Explorer pane, the user right-clicks the "Work" version of her project and selects the "Generate Version" option in the pertinent context menu.

This invokes a wizard that automates the creation of the scripts used to deploy the warehouse to the target system. The user specifies a version and selects a location for saving the output scripts. Optionally, clicking a button to validate the project tests the model against a battery of validation rules. biGENiUS lists any detected errors or warnings in a "Validation Results" pane, and a developer uses this as a starting point for troubleshooting.

Once the user clicks "Generate," biGENiUS creates the SQL scripts to instantiate the necessary data warehouse layers, e.g., cleansing area, landing zone, data model, and core structures. It also creates the Windows batch files (.CMD) used to schedule and execute SQL scripts on the warehouse target. Deploying the data warehouse is straightforward enough: a user double-clicks (or runs, via the Windows CMD shell) one batch file to deploy the data model and another batch file to trigger the ETL data flows that populate the warehouse.

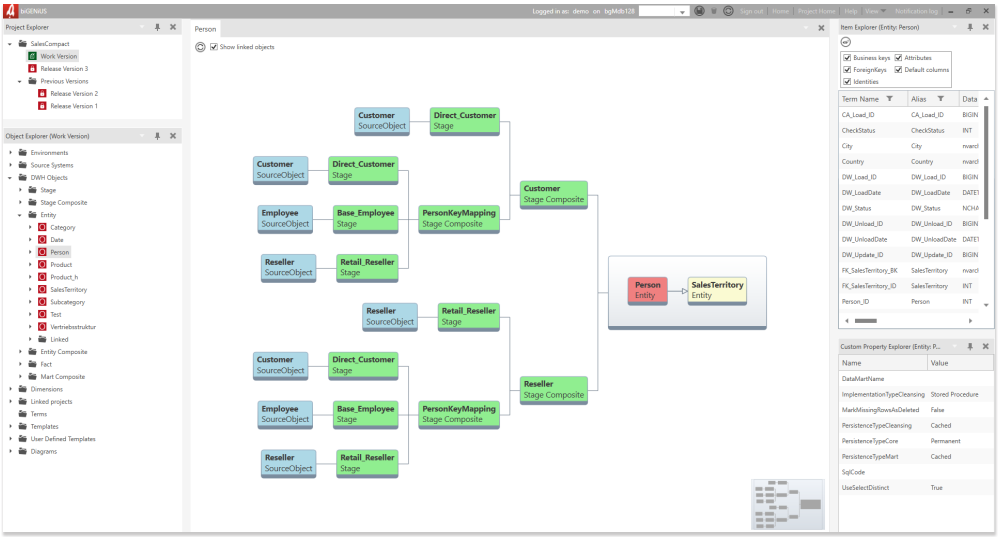


Like most DWA tools, biGENiUS automates the creation and curation of documentation. As part of the process of creating the .CMD and SQL scripts used to deploy and instantiate the warehouse, it automatically generates HTML documentation, too. biGENiUS annotates the SQL code it generates; this SQL is clean, readable, and well documented.

Custom-Fitting the Data Warehouse

Designing and building a data warehouse for the purposes of a demo is one thing; designing and building a *production* data warehouse to support decision making in a large organization is quite another. Different people in different roles have different responsibilities and expectations with respect to the kinds of decisions they need to make, along with the level of detail and the breadth of context they require to make them.

Fundamentally, the challenge is providing the right data at the right time to the right people. The complexity inherent in *scaling* decision support is what makes data warehouse development so costly and arduous. Reducing cost and complexity by simplifying development and maintenance is the *raison d'etre* of data warehouse automation.



biGENiUS does a good job of simplifying and, when practicable, automating … the important but thankless tasks that do little to stimulate human ingenuity or creativity.

Automation is only a relatively small part of this. The inconvenient truth is that the design, development, and ongoing maintenance of a production data warehouse system always requires the technological equivalent of custom tailoring. The logical data model that is created by the business — i.e., the business's representation of itself to itself — will almost always require significant customization. Reverse-engineering a data model from scratch *seems* simple enough. In practice, however, data-driven modeling is a time-consuming process that almost always requires cooperation among different stakeholders, such as data modelers, business subject-matter experts, DBAs, architects, etc. A DWA tool cannot eliminate this custom tailoring. Instead, the best DWA tools expose features (GUI shortcuts and user-driven wizards; drop-down menus; user-customizable fields) that simplify and accelerate this process. Automation, when practicable, is a bonus.

biGENiUS is no exception. In practice, the developer, modeler, architect, etc., uses the same shortcuts, wizards, and graphical modeling facilities described earlier to change or customize a data warehouse system. biGENiUS automates a number of tasks — e.g., discovery of source data structures; the identification of dimensions and, if applicable, the mapping of multiple attributes to a dimension; the generation and maintenance of documentation, metadata, and data lineage; the deployment of the data warehouse itself — that are time-consuming,

complex, or tedious. Basically, it does a good job of simplifying and, when practicable, automating most of the "scut" work — the important but thankless tasks that do little to stimulate human ingenuity or creativity.

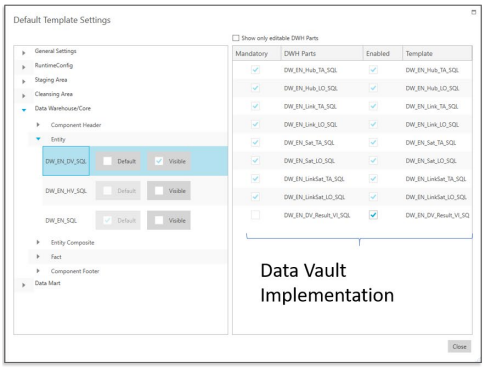
When a modeler defines an entity, for example, biGENiUS automatically generates an assortment of related cleansing, core, and data mart objects. In the process of custom-fitting the data warehouse, the user can drill down into these objects to make granular changes such as the following:

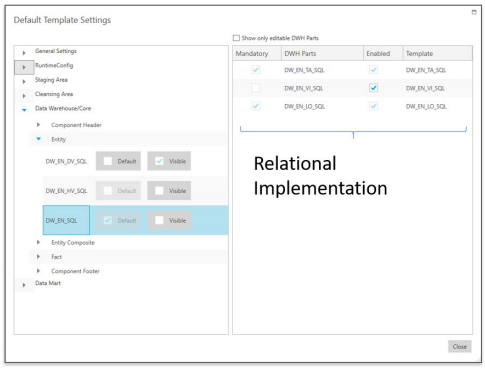
- Specifying business rules and business logic;
- Adding new attributes;
- Changing foreign keys, source mappings, etc.

If necessary, the user can drill down into the SQL code itself using biGENiUS's SQL source code editing features and make the requisite changes. If she changes the source template that is used to generate the object, biGENiUS will optionally incorporate her code into *all* objects of that class that it generates.

Or take another example, the process of moving from one schema — a classic 3NF warehouse — to another, such as a Data Vault 2.0. This is usually a non-trivial exercise, even with a DWA tool. However, biGENiUS supplies a number of pre-fab "Configurator" generators the modeler can use to rapidly generate different types of data models. She can use Configurators to convert from one type of data ingest paradigm (e.g., on-premises, batch ETL) to another (cloud data lake, event-based); from one type of data warehouse target (SQL Server) to another (Oracle), and also to change the core data warehouse data model, in this case converting from a 3NF data model to a Data Vault 2.0 model. The Configurator is basically a template that uses biGENiUS's metadata repository as an input source and generates a model with the appropriate output structures — in this case, the hubs, links, and satellites that are used in Data Vault 2.0 modeling.

Different implementation options based on different default template settings





To take another example, imagine an organization wants to move away from an RDBMS-based data staging model and implement Azure Data Lake instead. In biGENiUS, this actually becomes a fairly straightforward process: A developer, modeler, or architect would use the supplied Azure Data Lake Configurator to regenerate the organization's existing ETL stage data flows for the new data lake environment. biGENiUS provides Configurators to generate new data flows for a wide variety of sources, including Apache Kafka, the open source software stream processing engine. A developer can ingest data from Kafka into Azure Data Lake and automatically apply transformations before it is loaded into the warehouse staging area.

On its own, biGENiUS does not implement the equivalent of the distributed version control systems (VCS) used in conventional software development. But it can integrate with (and save project data such as documentation, data models, and source code files to) a distributed VCS such as Apache Subversion (SVN), Git, or Microsoft's Team Foundation Server (TFS).

biGENiUS's versioning feature permits the kind of code-test-and-iterate approach that is integral to test-driven software development.

In this scheme, biGENiUS itself takes care of metadata versioning; SVN, Git, or TFS serve as a repository for all project collateral. This approach permits a developer, modeler, etc., to design and build different instantiations of the data warehouse or related warehouse structures for different use cases such as sandbox, test-dev, or production, as well as to rapidly iterate as part of the process of building and customizing analytical deliverables for the business. For example, a developer customizes the SQL code in one of the templates biGENiUS uses to generate an object. She might not want this code to be used to generate all objects of this class. No problem: she can create a new version of the template and invoke it when necessary as an alternative to biGENiUS's default template.

Versioning helps accelerate the iterative process of designing and hardening a data model, too. IT can iterate through different versions of a model, enlisting business stakeholders to test sample reports, KPIs, or dashboards against live data. In this way, it is possible to rapidly build, test, modify, and improve models to address the requirements of different user roles, use cases, or privileged users. Iteration is likewise essential for trouble-shooting common data loading, mapping, and transformation issues. Finally, iteration also helps accelerate the process of building star/snowflake schema models or subject-specific data marts for consumption by downstream BI tools. biGENiUS's versioning feature permits the kind of code-test-and-iterate approach that is integral to test-driven software development.

Final Thoughts

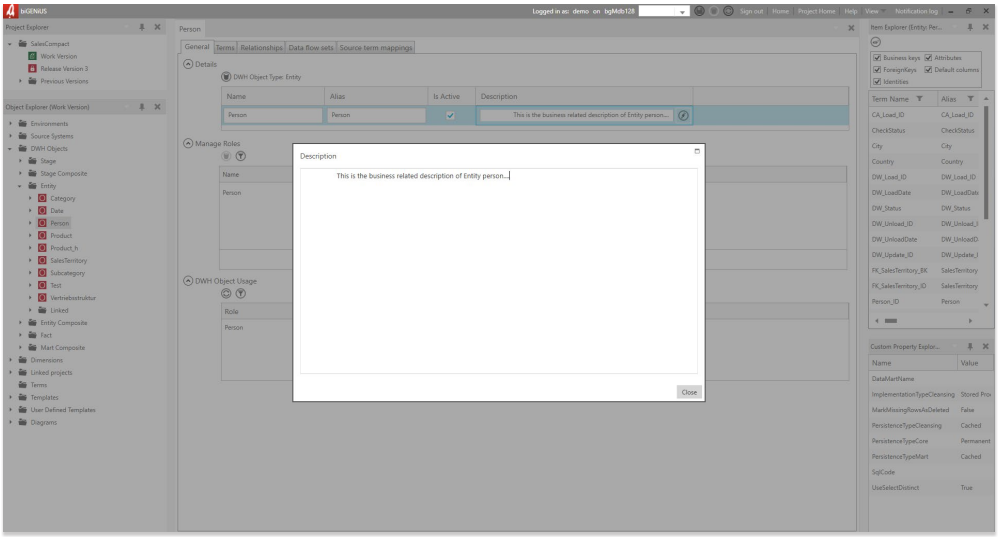
This walk-through relies on a number of ideal assumptions.

- Upstream sources are available, accessible, and well understood.
- People, process, and technology issues don't exist; organizational politics is not a serious impediment.
- The task of modeling, be it business- or data-driven, is simple and straightforward; the resultant models require only relatively minor customization.
- The generation and deployment of the warehouse is glitch-free.

Most important, this walk-through assumes not only that the business has an excellent understanding of itself and its operations, but also that the IT people designing the warehouse have an excellent understanding of the business, its operations, and how these operations relate or correspond to their own. It likewise assumes there is consensus at all levels of the organization that the proposed data warehouse project is aligned with or at least does not impinge upon their needs, priorities, and perquisites.

This never happens. In painful point of fact, this is why we have DWA tools.

This is where biGENiUS shines. Like all DWA tools, its UX is designed with IT users in mind. But its overall *usage* experience lends itself to and benefits from at least some involvement by the business. This might take the form of business collaborating in the design, development, and refinement of a logical data model; of the business participating in the iterative development and testing of the warehouse data model, presentation layer, and subject-specific data marts;

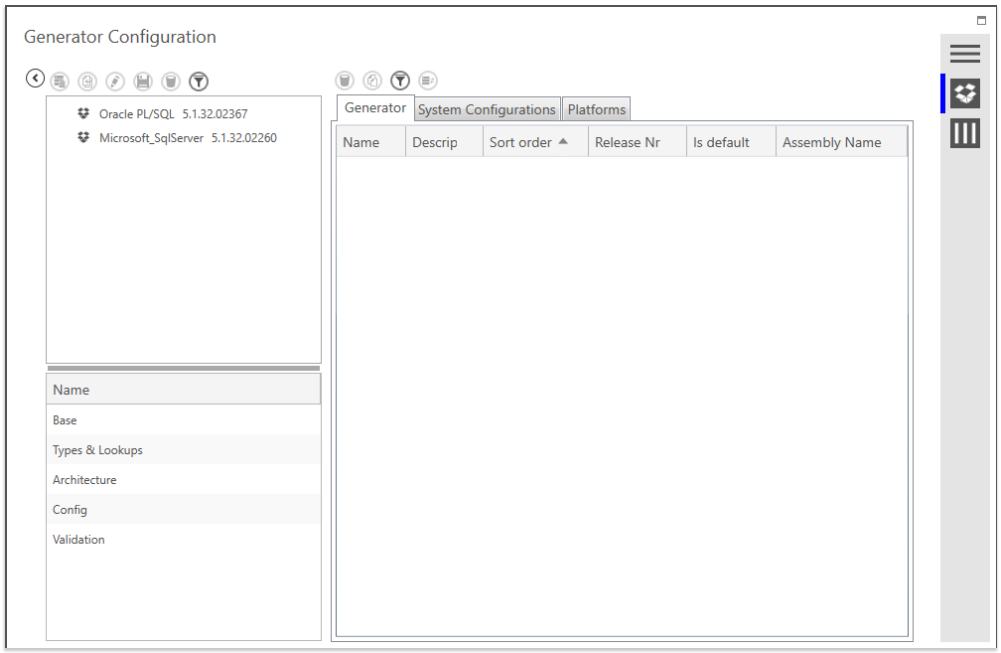


of its participation in a DataOps-like development regimen that enlists business and IT people in continuous development, testing, and delivery; or of something else altogether.

The best way to give the business the data warehouse system it wants and needs is to involve key business stakeholders in the diverse phases of designing, testing, and enhancing that system

The best way to give the business the data warehouse system it wants and needs is to involve key business stakeholders in the diverse phases of designing, testing, and enhancing that system. biGENiUS was not necessarily designed with this highly immersive business-driven development scheme in mind, but its overall UX should in practice permit (and perhaps even promote) useful collaboration between IT and the line of business. That is one hallmark of a good DWA tool.

Another is that biGENiUS's capabilities — and the design of the product itself — anticipate the analytical needs of tomorrow. For example, biGENiUS's "Configurator" generator templates allow it to accommodate not just any notional data source or deployment target, but alternatively, other conceivable data ingest and movement paradigms. This flexibility is the fruit of a design decision that permits a degree of abstraction between biGENiUS's metadata repository on one hand, and the specifics of data exchange APIs, data modeling conventions, and hosting context on the other.



In the biGENiUS model, moving away from a conventional (batch-oriented, data-at-rest) ingest paradigm to perhaps a staging area hosted in a cloud data lake that accommodates streaming ingest, in addition to conventional batch ETL, is a matter of queuing up the right Configurator script (e.g., Microsoft's Azure Data

Lake) and using it to regenerate an existing staging area. Next, it involves using biGENiUS to create new data flows for the appropriate sources: Kafka, HBase, etc. The approach is at once extensible and transparent: e.g., moving from one data warehouse target (Oracle, Teradata) to another (Azure SQL Data Warehouse) isn't a black-box proposition but accessible (and customizable) in the form of a Configurator generator template.

The product isn't perfect. Today, biGENiUS offers little in the way of built-in monitoring or management features. Managing the data warehouse and, consequently, performing certain diagnostic tasks such as trouble-shooting data loading errors will require the (IT) user to switch between biGENiUS and a database-specific management tool, such as SQL Server Management Studio. (biGENiUS says it plans to deliver an integrated GUI that incorporates manageability features at some point this year.) And even though its array of GUI shortcuts, wizards, and visualizations is helpful enough, biGENiUS is *not* a general-purpose self-service tool. Its ideal self-service user is an IT person or, alternately, an IT person who works collaboratively with business stakeholders. In the near term, this isn't necessarily a problem. After all, one big benefit of a DWA tool such as biGENiUS is that it allows IT to provision data much more quickly to accommodate self-service users and, moreover, to do so in the context of an analytical development process that is managed, repeatable, and governable.

biGENiUS's capabilities — and the design of the product itself — anticipate the analytical needs of tomorrow.

On the other hand, there's a strong case to be made that certain users could and should play a more constructive role in the process of analytical development, especially in conjunction with a system as important as the data warehouse. Today, self-service users commonly access and provision their own datasets for analysis; could they not use a DWA tool to do the same thing? At this point, an especially gifted self-service user could make productive use of biGENiUS; the average Tableau user could not.

This challenge is common to all DWA tools. biGENiUS, like competitive offerings, is a powerful self-service tool for *IT* users; will it evolve to accommodate other users? That is up to its parent company. For the present, Trivadis markets a useful and extensible DWA tool in biGENiUS: a product that is built for the analytical environments of today *and* tomorrow.

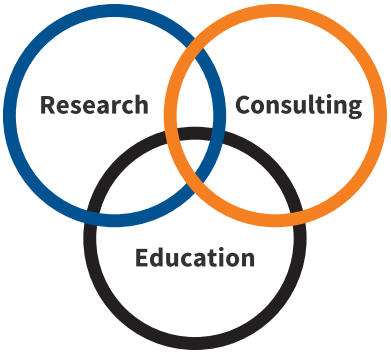
About the Author

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 biGENiUS

biGENiUS is part of Trivadis Group. Trivadis is an IT service company with 16 locations in Switzerland, Germany, Austria, Denmark and Romania with over 700 IT specialists in Data Management and Data Analytics. Together with customers and partners we shape the digital future and accompany our customers in their digital transformation.

With our Big Data and DW Automation Tool biGENiUS we provide highly automated support for the whole lifecycle of Data Analytics solutions both for traditional DW- and Big Data solutions. More than 250 successful projects and a rapidly growing partner network worldwide speak for the outstanding concepts in biGENiUS

Our Vision: To enable a world in which intelligent IT makes life and work easier as a matter of course.

Learn more