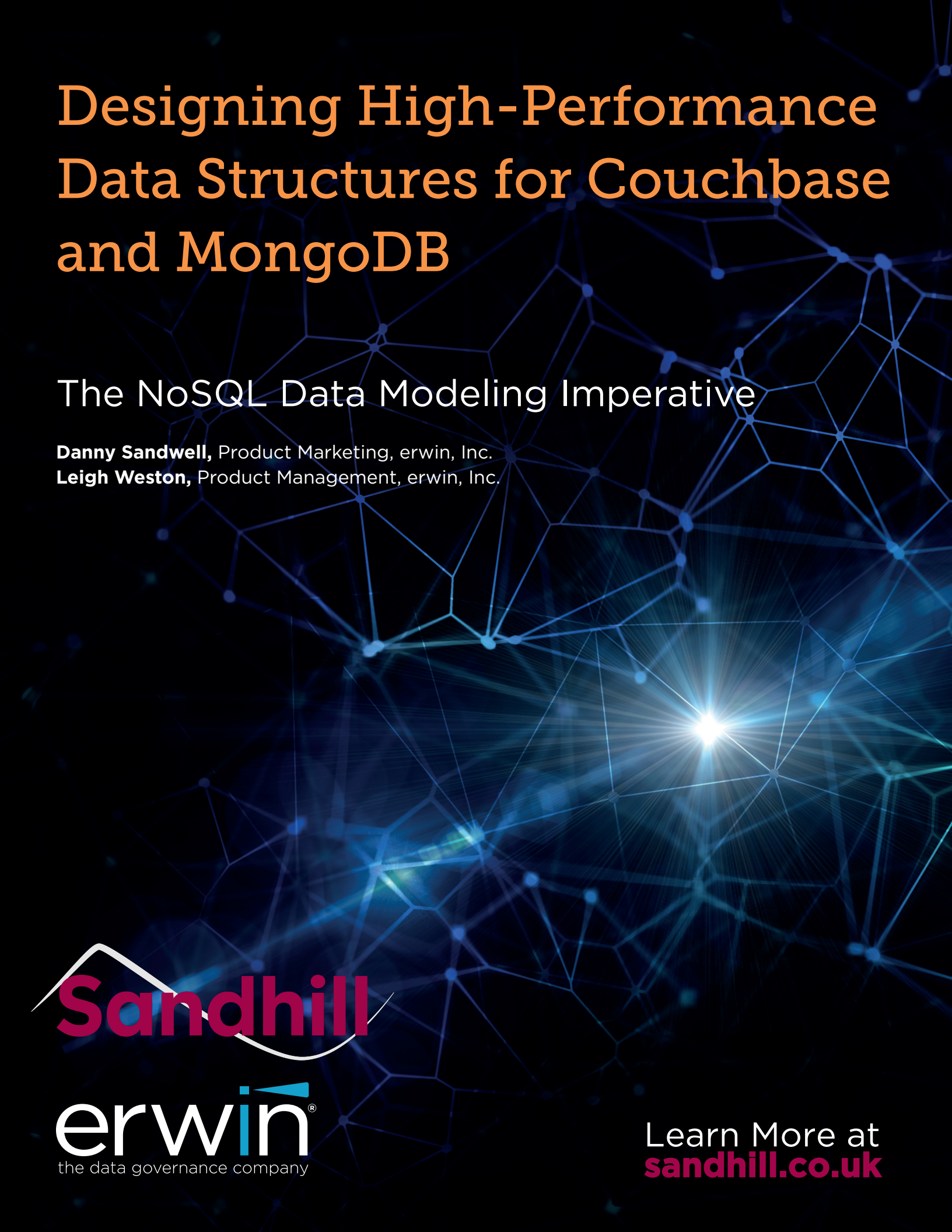
Designing High-Performance Data Structures for Couchbase and MongoDB

The NoSQL Data Modeling Imperative

Danny Sandwell, Product Marketing, erwin, Inc.
Leigh Weston, Product Management, erwin, Inc.



When Not "If" NoSQL

We've heard the business case and accepted the modern technology justifications for adopting NoSQL database management system (DBMS) solutions to support data-driven business transformation.

DBMS products based on rigid schema requirements impede our ability to fully realize business opportunities that can expand the depth and breadth of relevant data streams for conversion into actionable information. New, business-transforming use cases often involve variable data feeds, real-time or near-time processing and analytics requirements, and the scale to process large volumes of data. NoSQL databases, such as Couchbase and MongoDB, are purpose-built to handle the variety, velocity and volume of these new data use cases. Schema-less or dynamic schema capabilities, combined with increased processing speed and built-in scalability, make NoSQL the ideal platform.

Now the hard part. Once we've agreed to make the move to NoSQL, the next step is to identify the architectural and technological implications facing the folks tasked with building and maintaining these new mission-critical data sources and the applications they feed. As the data modeling industry leader, erwin has identified a critical success factor for the majority of organizations adopting a NoSQL platform like Couchbase and MongoDB.

Successfully leveraging this solution requires a significant paradigm shift in how we design NoSQL data structures and deploy the databases that manage them. But as with most technology requirements, we need to shield the business from the complexity and risk associated with this new approach. The business cares little for the technical distinctions of the underlying data management "black box." Business data is business data, with the main concerns being its veracity and value. Accountability, transparency, quality and reusability are required, regardless. Data needs to be trusted, so decisions can be made with confidence, based on facts. We need to embrace this paradigm shift, while ensuring it fits seamlessly into our existing data management practices as well as interactions with our partners within the business.

Therefore, the challenge of adopting NoSQL in an organization is two-fold: 1) mastering and managing this new technology and 2) integrating it into an expansive and complex infrastructure.

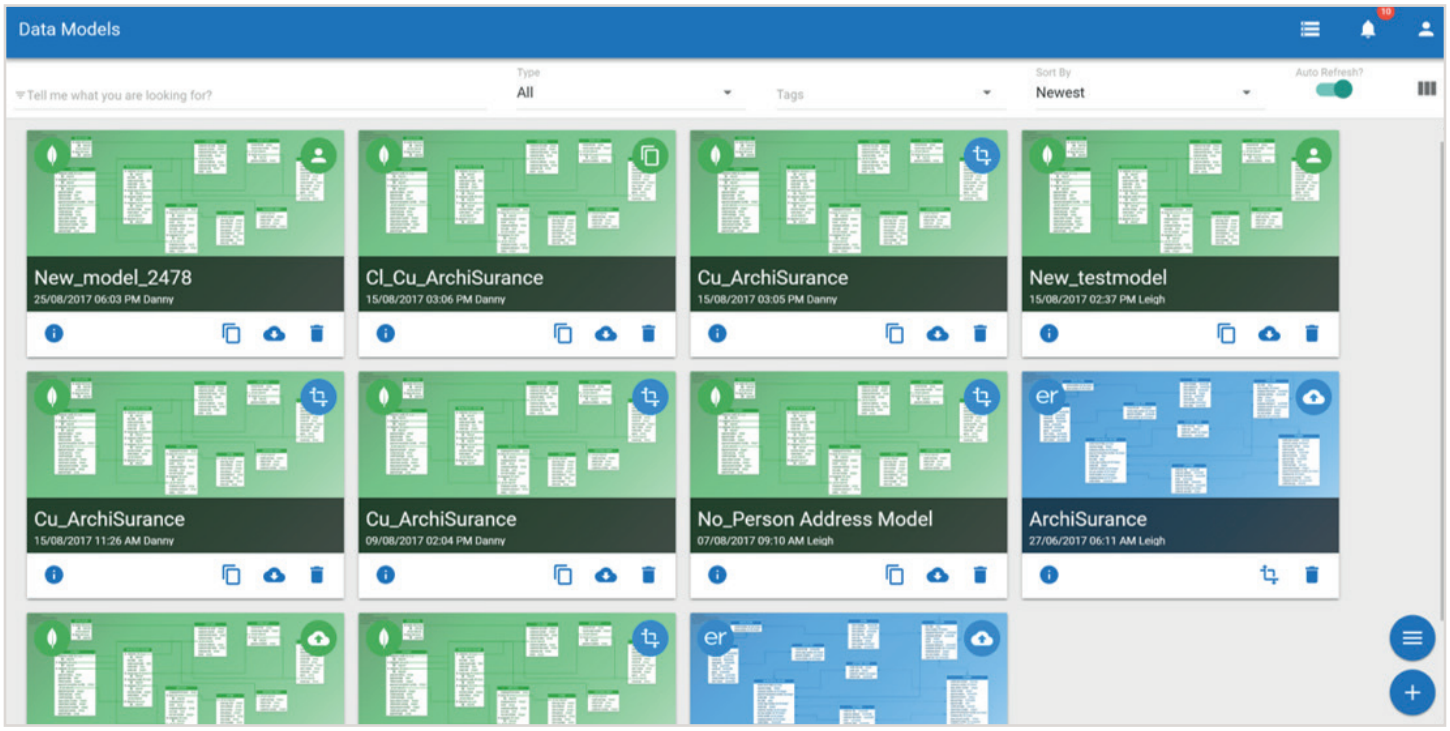
A Historical Perspective of Database Design

SQL (Structured Query Language) is a traditional programming language used to manage data in a relational database.

It has been widely adopted because it helps to maintain the referential integrity, constraints, normalization and structured access for data across disparate systems.

Traditional database design, focused on normalization and storage optimization, has been driven by the technological playing field of the day and the belief that normalization and the resulting referential integrity it provided was paramount. Relational database platforms thrived on this model that provides in-depth capabilities and mechanisms to manage data with integrity. The upside of this design is the reduction in complex application code to maintain the data. The downside is the loss of flexibility in data structures due to rigid schema requirements. Querying is also very complex, requiring lots of joins that results in less than stellar performance.

For the majority of historical data use cases, these were trade-offs we were willing to make because we could allocate more resources when query performance was paramount but real- and near-time performance requirements were not the norm. But when we look at new data use cases, development technologies and techniques, and the DBMS capabilities and behaviors they demand, the database design deck has been shuffled. The needs for real-time access and analysis drive demand for real-time performance. Therefore, query performance is now king from a design perspective. We have the new platforms, technologies and application developers to manage the risk of de-normalization, and we can now embrace it in pursuit of simplified querying and high-performance response.



Import erwin DM model, reverse engineer NoSQL instances, create new NoSQL models.

Query-Optimized Database Design

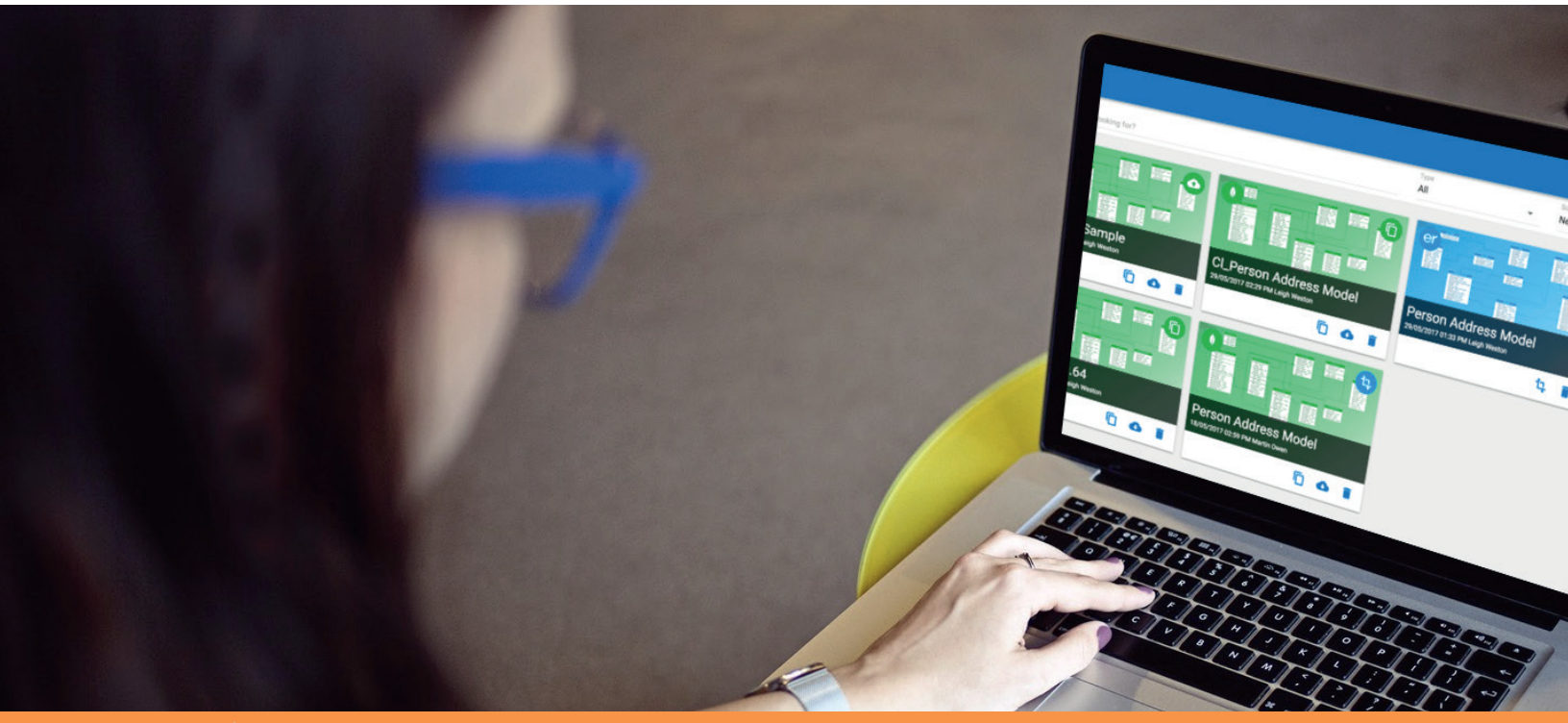
Couchbase and MongoDB are fast NoSQL databases, providing high performance with high availability.

They scale easily and offer rich query language in addition to flexible storage that does not require pre-defined tables with keys and relationships. Couchbase and MongoDB allow you to index attributes for performance with multiple, simultaneous database connections to increase throughput with asynchronous queueing. Unfortunately, NoSQL is not a cure for all your performance woes. A single complex query can bring your code to a grinding halt, and the design of the JSON-like documents and collections has a major impact on the performance of applications accessing Couchbase and MongoDB and their associated queries.

NoSQL design considerations and structures are radically different, even counterintuitive, and not what traditional database administrators and data modelers are used to. In the NoSQL world, there are no data normalization or storage rules. In fact, denormalization and duplications often are encouraged to make queries faster, making business operations more efficient. In addition, joins are rarely supported. They are instead the responsibility of the application code, so you need to consider this before structuring your data because application joins are expensive and will impact query times.

The net-net is that we used to design database schema based on what needs to be stored. The storage-optimized model ensured efficiency and integrity in data storage. With NoSQL, we now design data structures based on the questions we want the data to answer:
What do I want to achieve with this information? What queries typically need to be run? How frequently do I need to create, read update and/or delete data? New development technologies allow us to manage the risk to data integrity, so now we can structure data with the emphasis on simple queries and response speed.

Data modeling can guide you through this paradigm shift, enabling you to successfully incorporate modern database design into your data management practices and take the pain out of adopting NoSQL.



Tried-and-True Data Modeling

Data modeling is a rigorous discipline with numerous benefits:

Using a visual method of collecting data requirements, specifying and organizing them with proven notations, and collaborating around the model to ensure the data aligns to the business need. It allows data requirements analysis and design to be done in a rigorous, efficient and effective way by breaking down complexity, incorporating multiple perspectives, and documenting the details in a centralized and malleable fashion. Data modeling tools automate many time-consuming tasks, accelerating the delivery of value to the enterprise. With NoSQL, the motivations and resulting structures might be different, but the method and benefits of data modeling are largely the same.

VISUALIZATION AND DOCUMENTATION OF CONTENT AND STRUCTURE

Using a rigorous approach with a graphically rich visualization of business data requirements enables team members to understand, contribute and optimize business data sources. Specification, verification and organization of data elements is efficient and effective, ensuring completeness, connectivity and alignment with organizational standards.

With erwin Data Modeling (DM), you can use proven logical entity relationship diagrams (things, details about things, and relationships between things) to capture and organize business data requirements. With erwin DM NoSQL, you can depict these requirements in a native document format for deployment with NoSQL databases.

GUIDED DESIGN TRANSFORMATION

Leveraging data modeling to guide and execute structural transformation helps database designers to deploy and analyze multiple scenarios in a low-cost, low-risk, easy-to-consume fashion. Abstracting the design allows architects to try different structures, employ normalization and de-normalization rules, and better understand the costs and benefits of these optimization strategies long before any technology goes into production. This capability ensures that the application is ready for prime time.

At erwin, we provide three conversion options: 1) normalized that reflects a traditional SQL approach, 2) de-normalized with tables embedded as much as possible, and 3) custom that allows you to choose what to embed and what to make a reference. In all three cases, erwin guides you through the process and manages the mapping and construction of the resulting structural changes.

Tried-and-True Data Modeling

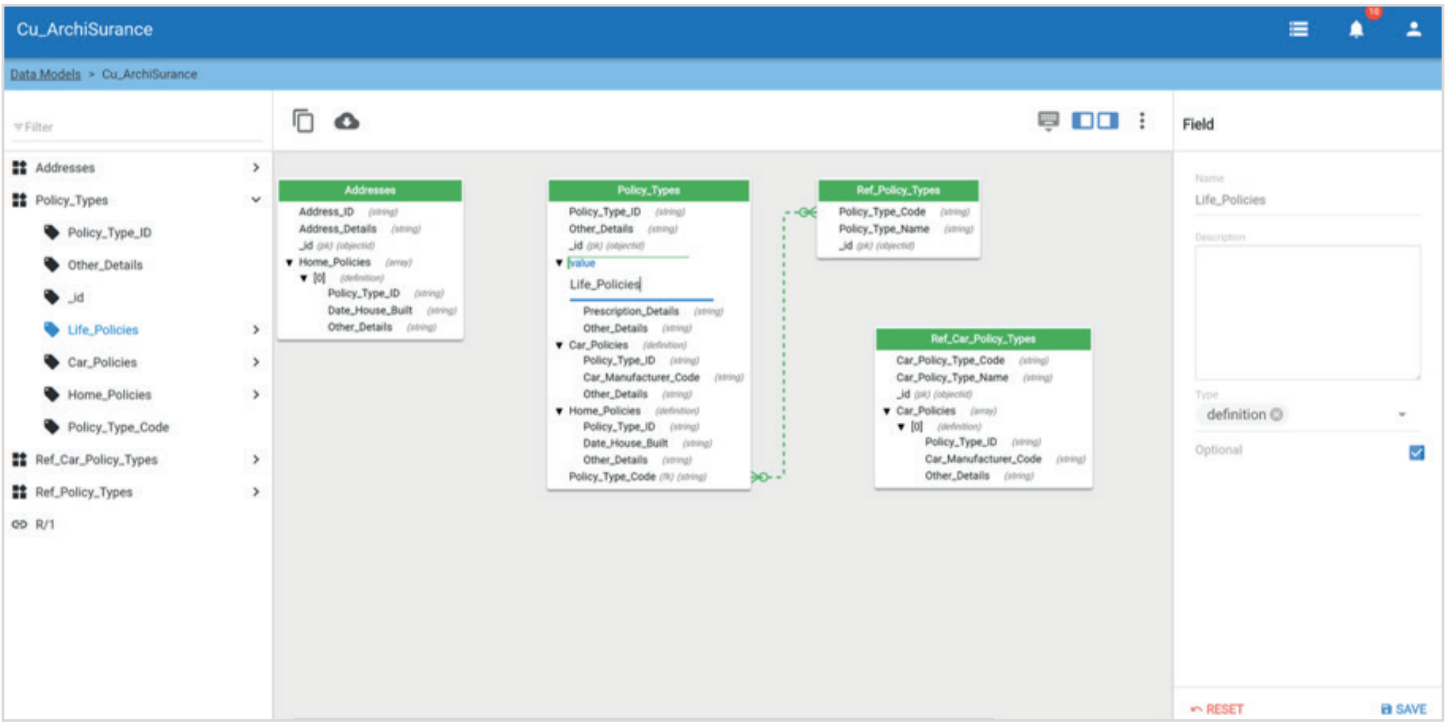
(continued)

TASK AUTOMATION FOR DATA DEFINITION AND DATABASE DEPLOYMENT

History has shown that a model-driven approach will not work unless there's sufficient automation for designing patterns and accelerating deployment tasks once the model is finished. The abilities to create models automatically by reading the DBMS catalog (reverse-engineering) and generate the structures from the model (forward engineering) are critical. Both model development and database deployment are then accelerated, saving countless hours of manual analysis, construction and coding. At erwin, we provide round-trip engineering for legacy databases, as well as Couchbase and MongoDB in the NoSQL world. We also automate the generation of test data (based on the model) to enable fast, effective test cases.

STAKEHOLDER COLLABORATION

These two proverbs continue to ring true: "It takes a village to raise a child," and "a picture is worth a thousand words." To ensure your application and the data it serves is ready for prime time, you need to incorporate feedback from a variety of technical and non-technical partners from within the business. A visual model is the best way to promote understanding and garner feedback in the development process. It's tough to grasp complex concepts from code or even pseudo-code, so having the ability to visualize a structure from inception through finish with multiple iterations in between is invaluable. erwin DM NoSQL provides five (5) reviewer licenses for each professional (data modeler) license, allowing you to give your team easy access to the models you create via URLs. Business stakeholders on the frontend and developers on the backend of the process can visualize the model and add comments, which are stored in the model. Our notification capability keeps everyone in sync and accelerates collaboration.



Create NoSQL models with collections, references, embedded arrays and detailed field definitions.

erwin DM NoSQL Use Cases

Organizations are transitioning to NoSQL and need data modeling capabilities to help them successfully adopt this modern database technology.

That's why erwin developed a new edition of erwin DM called erwin DM NoSQL. We want to help our customers take advantage of NoSQL, while maintaining the integrity, quality and governance of their underlying business-critical information.

erwin DM NoSQL takes the pain out of designing high-performance NoSQL structures. Our model-driven approach enables you to master and manage this new technology, seamlessly integrating it into existing processes and data management infrastructures.

Following are three use cases for erwin DM NoSQL with Query-Optimized Modeling™. The solution supports data modeling for Couchbase and MongoDB.

MIGRATING LEGACY DATABASES

- If you're developing a net-new application, collaborate with business stakeholders and subject-matter experts, and then use erwin DM to create a logical model identifying, specifying and organizing your business data requirements. Focus on entities, attributes, keys, business names, business rules and meaningful definitions. Export your erwin DM logical model using our XML format.
- If you're migrating a legacy database, export your erwin DM logical or physical model of the legacy database using our XML format. If no data model exists, use erwin DM to reverse-engineer a model for export from the database catalog or DDL script.
- Import the model into erwin DM NoSQL. You can browse the model and visualizations in native ERD form.
- Transform your erwin models into NoSQL models using our guided transformation options to create a new model with native JSON-like depictions of documents, collections, embedded arrays and reference field (implicit relationships).
- Clone the converted models to edit and document the model. You can investigate different scenarios by further manipulating the structures using our transformations, create additional documentation relationships, and modify name, data types, etc.
- Generate and export deployment scripts to create and populate these collections, including sample data and NoSQL validators, to start testing your queries.
- At any point during this process, you can collaborate with others, using the model comments and built-in notification system.
- Once you've the finalized the NoSQL model, you can freeze the edits to share with developers so they can use it in production system.

erwin DM NoSQL Use Cases

(continued)

DOCUMENTING AND OPTIMIZING PREVIOUSLY DEPLOYED DOCUMENTS AND COLLECTIONS

- Reverse-engineer from NoSQL instances on premise and in the cloud.
- Clone the converted models to edit and document the model. Enter definitions and tags (alternate names) for documents, collections and fields. You can investigate different scenarios, by manipulating the structures using our transformations, create additional documentation relationships and modify name, datatypes, optionality etc.
- Generate and export deployment scripts to create and populate these collections, including sample data and NoSQL validators, to start testing your queries.
- At any point during this process, you can collaborate with others, using the model comments and built-in notification system.
- Once you've finalized the NoSQL model, you can freeze the edits to share with developers so they can use it in the production system.

DEVELOPING NET-NEW DESIGNS FOR DEPLOYMENT

- Create a new, empty NoSQL model.
- Drag and drop new collections. A unique name will automatically be created, so use the properties panel to edit the name appropriately. Each collection automatically will contain an editable ID field.
- Drag and drop additional fields as required. Each field will be auto-named. Use the properties panel to edit the name, enter a description/definition, and select a data type from the drop- down menu. Specify optionality, and assign any tags.
- Create reference and/or embed selected fields as required.
- Generate and export deployment scripts to create and populate these collections, including sample data and NoSQL validators, to start testing your queries.
- At any point during this process, you can collaborate with others, using the model comments and built-in notification system.
- Once you've the finalized the NoSQL model, you can freeze the edits to share with developers so they can use it in the production system.

erwin, Inc. provides the only unified software platform combining data governance, enterprise architecture, business process and data modeling. Delivered as a SaaS solution, these technologies work together to unlock data as a strategic asset so all enterprise stakeholders can discover, understand, govern and socialize data to mitigate risk, improve organizational performance and accelerate growth. For more than 30 years, erwin has been the most trusted name in data modeling and its software foundational to mission-critical data programs in government agencies, leading financial institutions, retailers and healthcare companies around the world.

Connect with us at
sandhill.co.uk