InfoAdvisors

Is your Data Modeling Workflow Agile or Fragile?

Karen Lopez, InfoAdvisors

InfoAdvisors

www.datamodel.com
karenlopez@Infoadvisors.com
@datachick

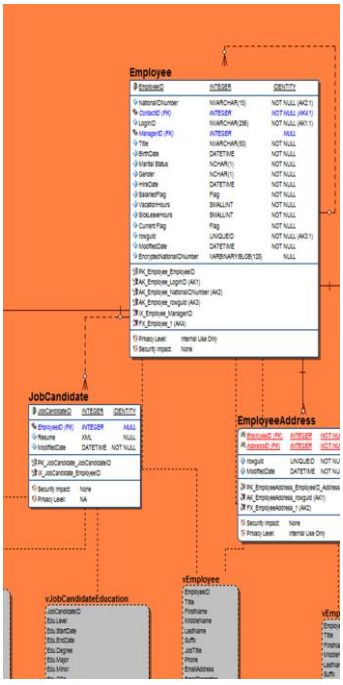
This white paper is sponsored by IDERA Inc.
ER/Studio is a registered trademark of Embarcadero Technologies, Inc.,
a wholly-owned subsidiary of IDERA Inc.

Data Modeling in an Agile IT Environment

Modern data modelers need to adapt processes to work on modern development projects.

Data modelers have access to a wealth of data value in the enterprise. They can bring that value to enterprise-class agile projects.

Leveraging data modeling and engineering tools is the perfect fit for agile values and principles.



Is your Data Modeling Workflow Agile or Fragile?

Introduction

Many data modelers are finding out that their traditional data modeling processes aren't a good fit for modern software development methods such as Agile or Scrum. Yet these approaches have become the *de facto* standard used by developers in enterprise organizations. Because we data professionals struggle to deliver value to these software projects, it's common for our products and services to be excluded from project work. Until we data professionals can tailor our approaches to meet the needs of software projects, we will continue to be left out of mission-critical projects.

The good news is that we *can* participate in these projects with some changes to our data modeling workflows. In this paper, we will describe the key components of a modern data modeling process.

Agile Overview

Agile approaches were developed in response to long, process-heavy, centrally-managed, death-march projects. A group of developers, exhausted and frustrated by not being able to develop software the way they wanted to, created The Manifesto for Agile Software Development (agilemanifesto.org). This manifesto has twelve principles, in summary:

- continuous delivery of valuable software
- welcome changing requirements, even late in development
- delivering working software frequently
- team working together daily throughout the project
- trusting the team to get the job done
- face-to-face conversation as the best way to get work done
- working software as the primary measure of progress
- maintaining a constant pace indefinitely
- focusing on technical excellence and good design
- simplicity
- use of self-organizing teams
- reflection, tuning and adjustments performed regularly

We recommend that you take a few minutes to read the full manifesto, even memorize it, as your agile team members will use these principles daily in making process and workflow decisions.

Scrum

Scrum (www.scrumalliance.org and scrum.org) is a type of agile method developed for teams with loosely defined requirements that change in the midst of software development, often several times for the same requirement set. Scrum, pulling from a rugby analogy, uses terms like *sprints*, *daily stand ups* (scrums), *backlogs*, *parking lots, product owners* and *scrum masters.*

Scrum has five values as a basis for its approach:

- Focus: dividing work into small efforts that can be accomplished in a relatively short period of time
- Courage: an emphasis on cross-functional team efforts
- Openness: regular open discussions about what's working, what isn't
- Commitment: due to empowerment, teams are motivated to take on challenges
- Respect: an emphasis on team respect

We also recommend you study the approaches, methods and terminology from Scrum development methods since they appear to be the most commonly used on enterprise projects.

There are other agile approaches such as XP, Kanban, Lean and more. This paper will use terminology from Scrum for simplicity purposes.

Fragile Development

Fragile development is the use of tools, methods, standards or deliverables that does not support the values and principles of agile development. There are some data management methods that may never be part of agile software development, but that's okay. Agile software projects will carry on whether or not they are part of the software process.

Data modelers who insist on forcing a traditional, process-heavy, massive deliverable base approach to agile projects will be seen as violating the values of agile *development*. That does not mean, however, that data modeling cannot be useful to agile projects. It can, and so can agile data modelers.

Agile Challenges for the Data Modeler

The agile movement has accomplished some great things in the IT world. There's a renewed focus on testing during development. Software is delivered more often and bugs are found much earlier in the process than older methods. Teams feel more empowered and users get the opportunity for hands-on work with application slices in weeks, not months. But along with these things, the traditional data modeler can find themselves scrambling to fit in.

It's All About the Software

One of the most difficult challenges for a data modeler is found in the core mission and value statements for agile: it's a **software** development approach. This means that the methods and values are focused primarily on the development process, with the assumption that infrastructure, integration with other systems, reference data, data life cycle, data sovereignty, etc. are all dealt with externally. What team members want from data modelers is a database, specifically one designed and ready with reference data and test data at the start of each sprint so that they can begin coding.

Another key aspect as a software process is that agile teammates value working code over all else. For an agile data modeler, working code is a functional database. If you are a logical-only data modeler, you can provide value by sharing your logical models, but you will not be a member of the agile team. Agile data modelers deliver code (full database scripts, alter scripts, reference data scripts and more) to the team. We also deliver beautiful data models for use in helping teammates understand the database. However, it's the code that is the primary measure of progress on an agile team.

No Specialists Allowed

One of the extensions to agile methods requires that all team members be able to perform any job on the project – in other words, everyone is a generalist. This means that there are no specialists in security, data modeling, databases or networking. This, again, is due to the fact that agile is primarily a software development approach. In reality, on enterprise projects, specialists are needed on the project. But it's likely that due to all the other challenges listed here, data modelers are the least likely to be invited to the party.

Documentation is a Dirty Word

Agile approaches call for "just enough" documentation. Many people who last saw a data model as part of a college homework assignment understand data models as something created as part of a checklist of documentation for software they developed. They produced it after they built some tables for their software. At most, they see data models as something that is reverse engineered from a database. They do not understand that data models are the record of all discussions, security needs, privacy needs, business requirements and design decisions made for a project or multiple projects.

This disconnect is the leading cause for agile teams to see data models as process-heavy documentation, not a living artifact for using model-driven engineering methods.

Stories as a Requirement

Most sprint planning provides the loosely defined requirements, as *stories*, at the start of each sprint. These stories were created by working with the product owner and a team member acting as a business analyst. This means that modelers have no time to ask follow up questions, to clarify vague requirements such as "each transaction must charge sales tax" or "we add a customer if they don't exist yet in the system".

Sprint Planning

Sprints are 2-3 week efforts to deliver a working piece of the system. Often sprint planning calls for work to be done in order based on priorities for the system being delivered first. In a point of sale system, planning may call for *Sell a Retail Item* to be the first sprint. For a developer, that involves recording data about a customer, item, price, promotion and sales tax. To a data modeler, that's weeks, if not months, of work to build all those entities and attributes. The scope of the story is very large for the database, but relatively small for the code that is just recording the fact that this customer bought this item at this price.

Sprint Timing

A typical sprint (or set of sprints) has sprint planning deliver the stories and backlog items at the beginning of each sprint to the entire team, including data modelers. That leaves the data modeler scrambling to understand the requirements at the same time the developers are waiting for the database. Teammates are left waiting for a functional database to start their coding activities while the data modeler is trying to understand the requirements and model dozens of entities and then build a physical model. As seen in *Figure 1 - Typical Fragile Sprint*, everyone gets frustrated with this process. Repeat it enough times and it becomes clear that the data modeler will be perceived as not being agile enough to be part of the team.

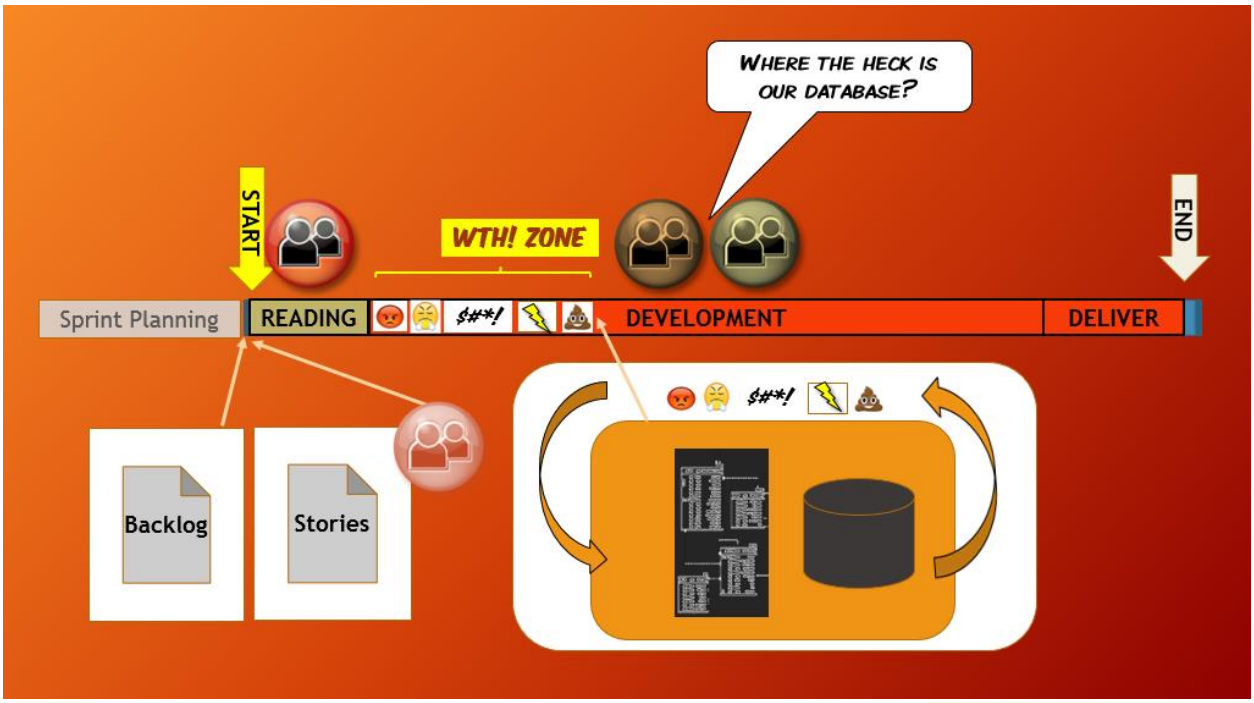


Figure 1 - Typical Fragile Sprint

Using an Agile, Not Fragile, Data Modeling Workflow

With all these challenges, how can a data modeler ensure his or her data modeling methods deliver value to agile teams? First, data modelers must understand the values and process embraced by agile teams and adjust accordingly. Second, data modelers must reflect those values while educating teams on the challenges we face in adjusting a single tightly integrated database design into a software process that focuses on dividing work into process-driven efforts. Finally, we must prepare our data modeling environments to support rapid and continuous delivery of database code.

The Right Sprint Workflow for Agile Data Modelers

The first thing that data modelers need to accomplish is to become part of the sprint planning process, due to the challenges mentioned above. As seen in *Figure 2 - Data Models & Sprint Planning,* data modeling for requirements has been moved to the sprint planning phase so that data modelers can be part of the development of stories and have access to the product owner to ensure critical data requirements are understood and ready for database design. By moving data modeling to the planning phase, modelers will deliver higher quality data models and therefore higher quality database designs, at the right time for the team.

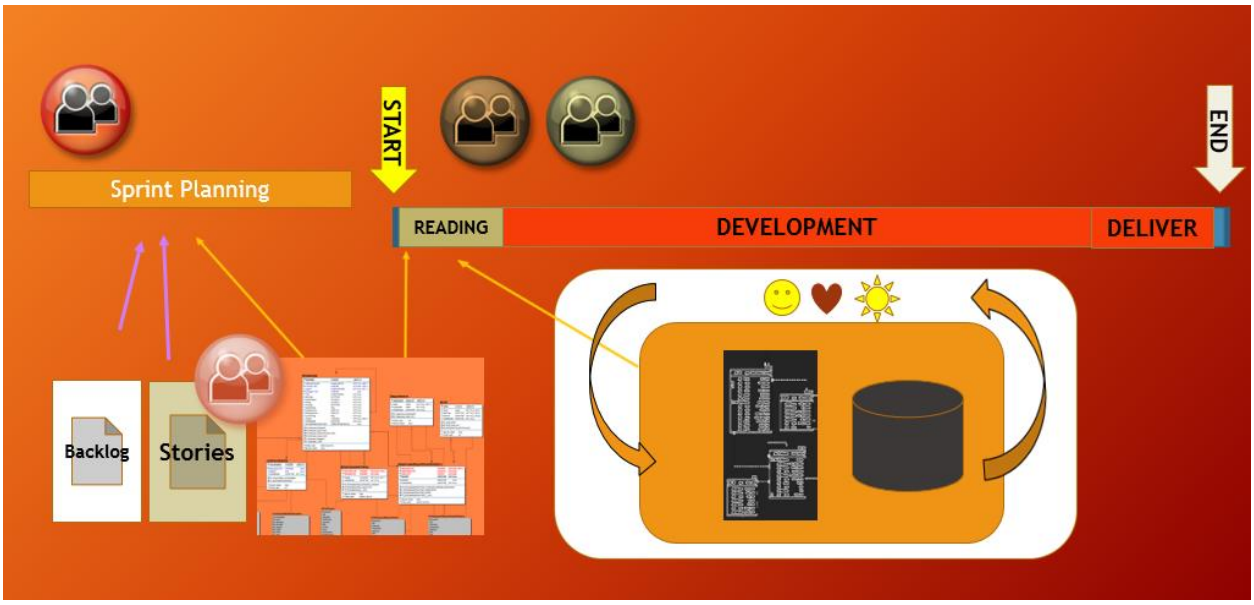


Figure 2 - Data Models & Sprint Planning

But not all sprint planning happens just before a sprint starts. It's common for teams to do sprint planning two or three sprints ahead of when a sprint starts. So instead of the data model being prepared just before the start, it's part of sprint planning several weeks (two to three sprints) ahead of when developers need a functioning database. This approach is the most ideal situation for the team.

Hands On with Databases

As mentioned above, agile approaches are focused on software delivery. For data professionals, this means databases. However, many data modelers have let their database skills fall behind current RDBMS (and other DBMSs) versions.

It's critical that an agile data modeler be able to engineer a database, even one with only a first cut design, on a regular basis.

This means having database client tools and drivers, having permissions for the right databases, and being ready to connect to them.

Data modelers likely won't need to develop database queries that will be deployed with the system, but they should have tools and skills to look at data in the database as well as forward and reverse engineer database objects.

Understand Changes to the Data Model… and Database

Iterative development means that changes to the models and databases will happen weekly or even more often. Data modelers will need to master their tool features to do fast database forward engineering, reverse engineering and compare and merge tasks.

While agile embraces change, an accidental or unplanned change can set the team back by days or weeks. The key to planned change is comparing versions of models and databases to understand the exact nature of each iteration of the database design. We do this level of change management via Model-Driven Database Design as seen in *Figure 3 - Model-Driven Database Design.*

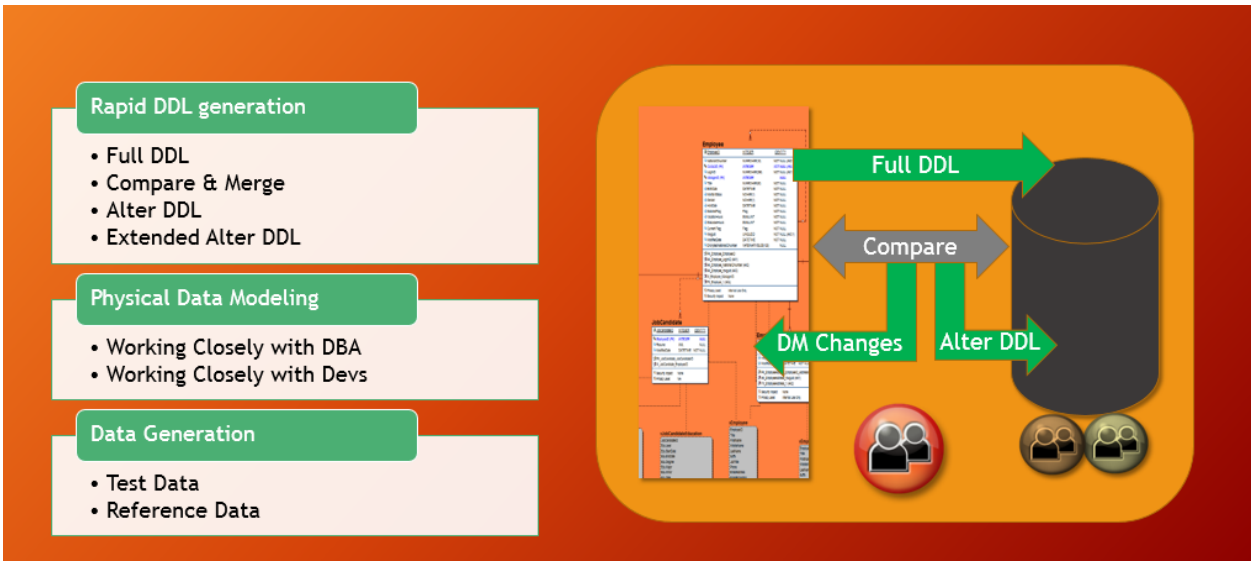


Figure 3 - Model-Driven Database Design

This concept is so important that we recommend data modelers get formal training on the compare features of their tool, then practice performing these tasks before joining an agile team.

Agile Data Model Documentation

While agile methods discourage the production of mounds of hand-written documentation, providing "just the right amount" of documentation does include publishing data models, metadata and data model diagrams where all team members can access them.

Providing searchable, commentable, browser-based access to all of these is the ultimate in agile documentation. This gives team members "self-service data models" and frees up data modelers to do core data modeling activities.

Tasks & Issue Management Tracking

While agile teams embrace process-light methods, they do make use of control mechanisms around changes, versioning and collaborative development. This often includes issue or bug management systems that allow team members to check in, check out and describe changes. They may also use a change control process such as tying every change in code to a specific task or issue (bug).

Data modelers must use a data modeling tool repository to manage change control, with compare features to ensure we understand the exact nature of changes, down to the finest granularity possible. We also need a way to tie these changes to tasks and external issue management items. One way to do that is right inside our data modeling tools, as seen in *Figure 4 - ER/Studio Tasks* and *Figure 5 - ER/Studio Change Records.*

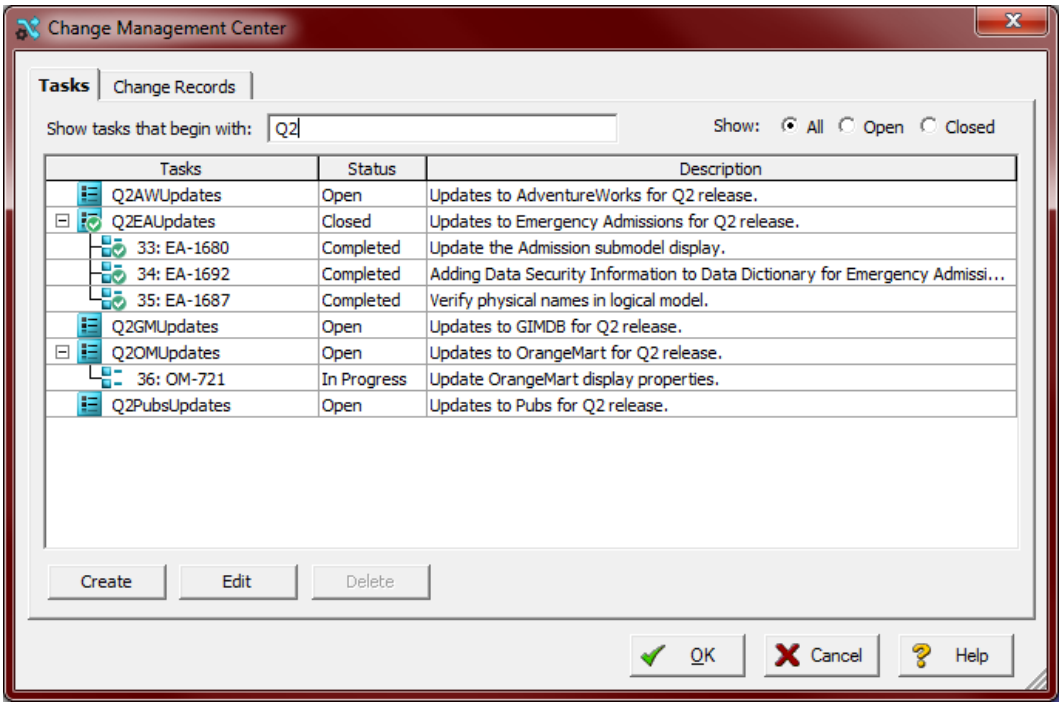


Figure 4 - ER/Studio Tasks

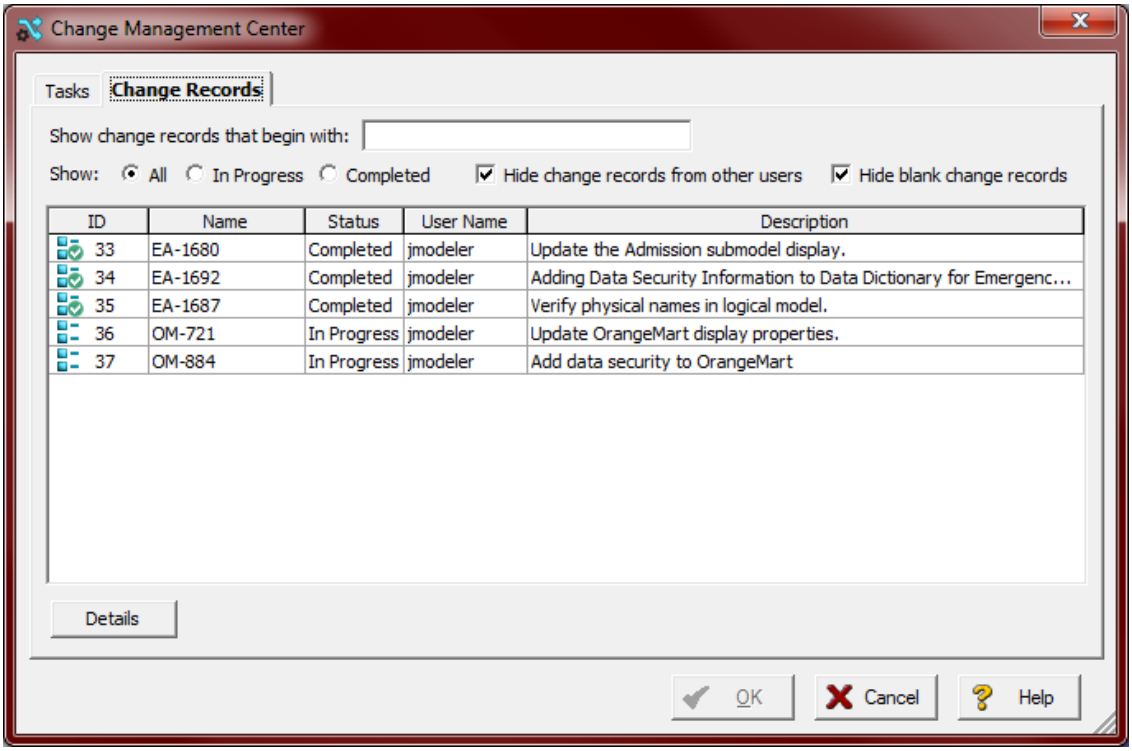


Figure 5 - ER/Studio Change Records

Having this level of change management allows multiple people to work on models, while having the context of why and how something changed in the data models.

Ten Tips for Agile Data Modelers

If you'd like to be a valued member of an agile team you should follow these tips to ensure you are ready for modern software development approaches.

1. Stop using the word *Documentation;* say **data models**.
2. Get agile and Scrum **training**. Get certified even.
3. **Prepare** your Agile Data Modeling environment.
4. Learn, practice and test your **Data Model iterations** processes and standards.
5. Learn to **automate** as much as possible.
6. Get data models and DDL tasks moved sprints ahead. At least **2 sprints ahead**.
7. Don't get pushed into **sprinting a marathon.**
8. Don't back off from Agile teams, even if they are hostile to data modeling…or modelers.
9. Don't be a **roadblock**. Get ahead of the sprints.
10. Use the same processes for **issue and change management** as the team uses.

About the Author

Karen López is Senior Project Manager and Architect at InfoAdvisors. She has more than twenty years of experience in helping organizations implement large, multi-project programs.

InfoAdvisors is a Toronto-based data management consulting firm. We specialize in the practical application of data management. Our philosophy is based on assessing the cost, benefit, and risk of any technique to meet the specific needs of our client organizations.

We want you to love your data.

Find us at datamodel.com

About IDERA Inc.

IDERA understands that IT doesn't run on the network – it runs on the data and databases that power your business. That's why we design our products with the database as the nucleus of your IT universe.

Our database lifecycle management solutions allow database and IT professionals to design, monitor and manage data systems with complete confidence, whether in the cloud or on-premises.

ER/Studio is the collaborative data modeling solution for data professionals to map and manage data and metadata for multiple platforms in a business-driven enterprise data architecture.

Whatever your need, IDERA has a solution.