ASSESSING SUPPORT FOR TOP-DOWN DATA MODELING IN TWO LEADING TOOLS

BY GEORGE MCGEACHIE

TABLE OF CONTENTS

INTRODUCTION

According to the Data Management Body of Knowledge (DMBOK) data modeling is “the process of discovering, analyzing, and scoping data requirements, and then representing and communicating these data requirements in a precise form called the data model” (DAMA International, 2017). Traditionally, data models have been constructed during projects that develop IT systems and databases to solve business problems. Many of these projects operate in isolation from other modeling efforts in the organization (I refer to this as ‘Silo Design’), whereas other projects operate as part of an enterprise-wide approach using a common data model from which their project models are derived.

In this review I examine two major data modeling tools - Idera ER/Studio Data Architect 18.5 (ER/Studio) and erwin Data Modeler Standard Edition 2020 R2 (erwin DM).

BASIC PRINCIPLES

Several types of data models are commonly produced, the principal ones being Conceptual, Logical, and Physical Data Models. The Conceptual Data Model especially is subject to a great deal of debate about how it is used and what it should contain. By comparison, there is broad agreement about the purpose and content of Logical and Physical Data Models and also dimensional models which are, in these tools, an extension of the Physical Data Models.

In the first edition of “Data Modeling Made Simple” (Hoberman, Data Modeling Made Simple, 1st Edition, 2005), Steve Hoberman describes the distinction between Logical Data Models and Physical Data Models simply and unambiguously - “the logical data model represents the rules behind how the business or application works independent of context. The physical data model represents the rules behind how the business or application is optimized for a specific context.” In Steve’s later book “Data Model Scorecard” (Hoberman, Data Model Scorecard, 2015) he goes on to say that the “logical data model looks the same regardless of whether we are implementing in MongoDB or Oracle”.

I agree with Steve, the logical view of data as expressed in the Logical Data Model (LDM) must not be influenced by my implementation decisions – these must be represented in the Physical Data Model (PDM). In Graeme Simsion’s research into actual data modeling practice he found, for example, that “specifying how subtype hierarchies will be implemented” (Simsion, 2007) is most commonly regarded as Physical Data Modeling.

This review is focused on the support provided for the production of Logical and Physical Data Models, maintaining the separation of the Logical and Physical views of data, and the support for tracing the links between the two models.

Both tools in this review can be used stand-alone or with a repository; the repository provides more than just a place to store models, it provides additional useful capabilities such as version control. Both tools also have web portals although the erwin product is a third party offering not produced by erwin.

The review focuses on using the two tools stand-alone (without the repository), examining their support for the two data modeling approaches mentioned above, making note of any relevant additional features that are available if you use the Repository. This is not unrealistic – on the projects where I used erwin DM or ER/Studio in the past, we did not have access to a Repository.

PRODUCT OVERVIEWS

ER/Studio and erwin DM are long-standing players in the data modeling tools market, focusing completely on the production of LDMs and PDMs. Both tools have powerful utilities for reconciling the differences between two data models (in this case the term 'data model' can include an SQL file, which the tool will use to construct a model for comparison purposes). These utilities provide the essential capabilities necessary for managing the links between data models that I explore in this review. Both tools allow you to choose the types of objects and properties to compare, and which changes to apply to either of the two models (a bi-directional merge). Also, both tools allow you to use subsets of the model (known by ER/Studio as 'submodels' and by erwin DM as 'subject areas') to define the scope for comparison.

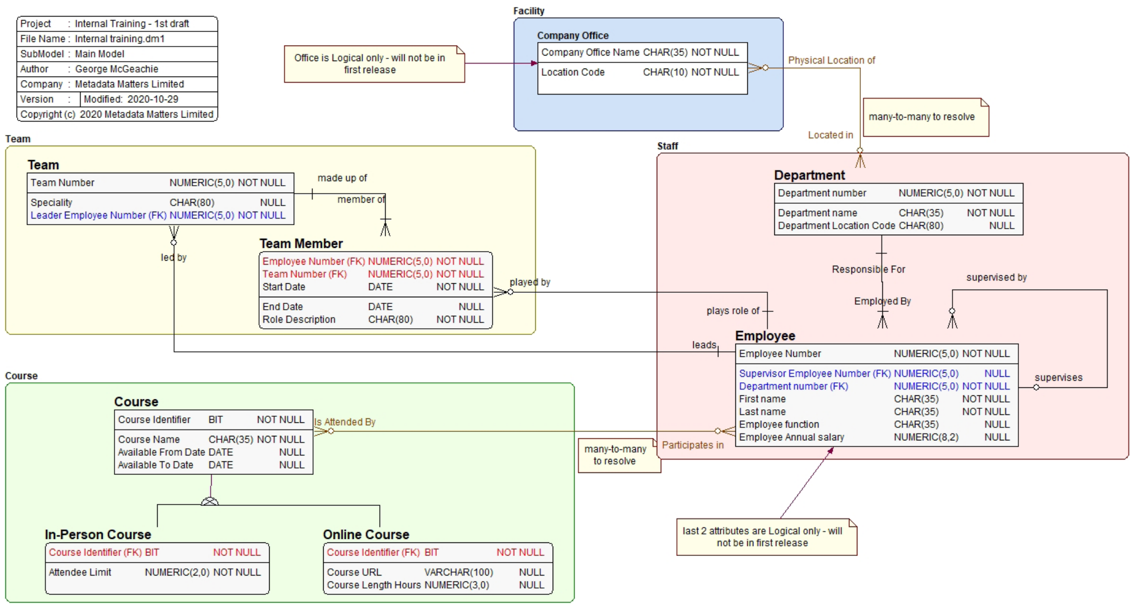
Erwin's comparison utility is called Complete Compare, while ER/Studio's is called the Compare and Merge utility. Detailed analysis of the capabilities of these tools is outside the scope of this review. The Design Layers feature of erwin DM appears to use a variation on the standard Complete Compare function that has access to more information about the links between objects.

Both tools include a feature called 'where used' which allows you to see a list of objects linked to the current object – in ER/Studio this can include objects in a different model, which is an important feature as we describe later, linked using Universal Mappings.

A huge difference between the two products is that erwin does not allow multiple physical models in the same overall model. This to me is a vital issue with erwin and obvious advantage for ER/Studio. ER/studio allows more than one PDM in the same file, and they can all be radically different from the LDM if needed. The 'combined' model in erwin DM cannot support multiple PDMs in one file, insisting that the PDM and LDM are two views of the same set of objects. We'll see this in more detail later on.

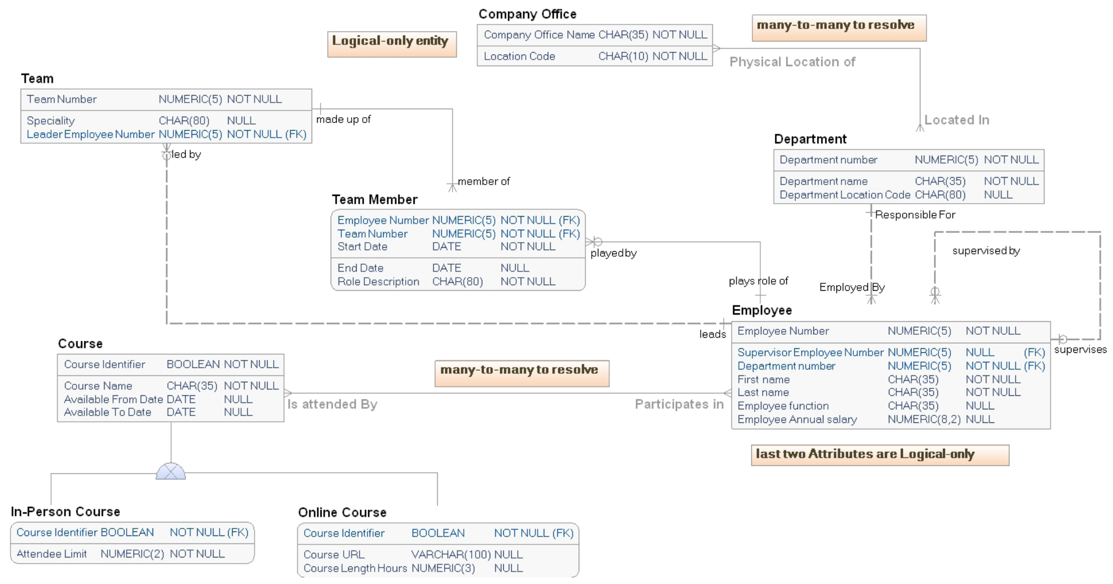
METHODOLOGY

I created a simple LDM in both tools with enough complexity to make sure that my comparison is realistic. One of the Entities (Company Office) and two of the Attributes (the last two Attributes in Employee) are marked as Logical-only so they will not be in the first version of a PDM. The model contains two many-to-many relationships that need to be resolved when the PDM is created.



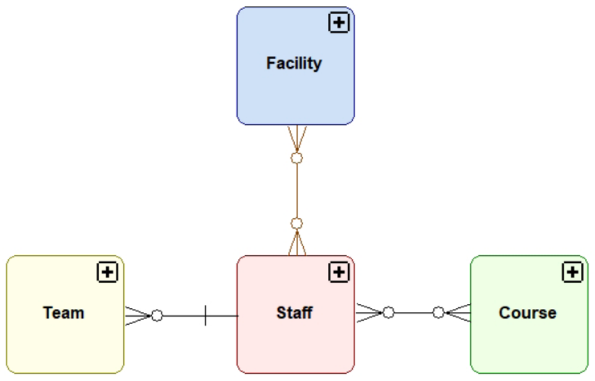
Project : Internal Training - 1st draft
File Name : Internal training.dm1
SubModel : Main Model
Author : George McGeachie
Company : Metadata Matters Limited
Version : Modified: 2020-10-29
Copyright (c) 2020 Metadata Matters Limited
Facility
Company Office
Company Office Name CHAR(35) NOT NULL
Location Code CHAR(10) NOT NULL
Office is Logical only - will not be in first release
Physical Location of
many-to-many to resolve
Located in
Team
Team Number NUMERIC(5,0) NOT NULL
Speciality CHAR(80) NULL
Leader Employee Number (FK) NUMERIC(5,0) NOT NULL
made up of
member of
led by
Team Member
Employee Number (FK) NUMERIC(5,0) NOT NULL
Team Number (FK) NUMERIC(5,0) NOT NULL
Start Date DATE NOT NULL
End Date DATE NULL
Role Description CHAR(80) NOT NULL
played by
Staff
Department
Department number NUMERIC(5,0) NOT NULL
Department name CHAR(35) NOT NULL
Department Location Code CHAR(80) NULL
Responsible For
supervised by
Employed By
plays role of
leads
Employee
Employee Number NUMERIC(5,0) NOT NULL
Supervisor Employee Number (FK) NUMERIC(5,0) NULL
Department number (FK) NUMERIC(5,0) NOT NULL
First name CHAR(35) NOT NULL
Last name CHAR(35) NOT NULL
Employee function CHAR(35) NULL
Employee Annual salary NUMERIC(8,2) NULL
supervises
Course
Course Identifier BIT NOT NULL
Course Name CHAR(35) NOT NULL
Available From Date DATE NULL
Available To Date DATE NULL
Is Attended By
many-to-many to resolve
Participates in
In-Person Course
Course Identifier (FK) BIT NOT NULL
Attendee Limit NUMERIC(2,0) NOT NULL
Online Course
Course Identifier (FK) BIT NOT NULL
Course URL VARCHAR(100) NULL
Course Length Hours NUMERIC(3,0) NULL
last 2 attributes are Logical only - will not be in first release

Here is the same model defined in erwin DM:



Company Office
Company Office Name CHAR(35) NOT NULL
Location Code CHAR(10) NOT NULL
Logical-only entity
many-to-many to resolve
Physical Location of
Located In
Team
Team Number NUMERIC(5) NOT NULL
Speciality CHAR(80) NULL
Leader Employee Number NUMERIC(5) NOT NULL (FK)
made up of
led by
member of
Team Member
Employee Number NUMERIC(5) NOT NULL (FK)
Team Number NUMERIC(5) NOT NULL (FK)
Start Date DATE NOT NULL
End Date DATE NULL
Role Description CHAR(80) NOT NULL
playedby
Department
Department number NUMERIC(5) NOT NULL
Department name CHAR(35) NOT NULL
Department Location Code CHAR(80) NULL
Responsible For
supervised by
plays role of
Employed By
Employee
Employee Number NUMERIC(5) NOT NULL
Supervisor Employee Number NUMERIC(5) NULL (FK)
Department number NUMERIC(5) NOT NULL (FK)
First name CHAR(35) NOT NULL
Last name CHAR(35) NOT NULL
Employee function CHAR(35) NULL
Employee Annual salary NUMERIC(8,2) NULL
supervises
leads
Course
Course Identifier BOOLEAN NOT NULL
Course Name CHAR(35) NOT NULL
Available From Date DATE NULL
Available To Date DATE NULL
many-to-many to resolve
Is attended By
Participates in
last two Attributes are Logical-only
In-Person Course
Course Identifier BOOLEAN NOT NULL (FK)
Attendee Limit NUMERIC(2) NOT NULL
Online Course
Course Identifier BOOLEAN NOT NULL (FK)
Course URL VARCHAR(100) NULL
Course Length Hours NUMERIC(3) NULL

You can see from these diagrams that the two tools produce similar objects and diagram symbols, though the ER/Studio diagram contains some big colored boxes that the erwin DM diagram does not include. These are “Business Data Objects” (BDOs), which can be used in a variety of ways.



In this model, I use them to represent high-level business concepts. Business Data Objects (BDOs) are the building blocks of Conceptual Data Models which aren’t supported at all by erwin. In this diagram, the BDOs are ‘collapsed’ so the entities are hidden, and only the key connecting relationships are shown.

I have used a few different icons to represent my opinions as I make my notes:

↓	Highlights an important point
☑	A potential advantage for this tool
☒	A potential disadvantage for this tool

TWO MODELING APPROACHES

In both of the following approaches, I need a Logical Data Model and two Physical Data Models – to create two very similar databases, implemented in PostgreSQL and Snowflake.

SILO DESIGN

The modeling effort is isolated from other modeling efforts in the organization, producing a Logical Data Model that will probably be inconsistent with Logical Data Models produced elsewhere in the organization. This is the approach that is usually described by erwin in their marketing material.

In both tools, this approach can be supported by using a single file, stored locally or in the central repository, that contains the required models.

ELDM-DRIVEN MODELING

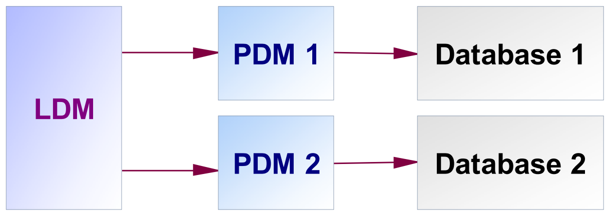
In this approach, there is a central Enterprise Logical Data Model (ELDM) from which project models are derived. Project models may be used to extend and alter the ELDM. Various other labels might be used for this central model, such as “canonical”. It may also be expressed as an ontology or something else that is not a Logical Data Model. In this review, I assume that it is indeed a Logical Data Model.

Both tools make it possible to create and manage a set of related models, including multiple levels of both Logical and Physical Data Models. Erwin DM supports this approach using ‘Design layers’. ER/Studio supports this approach using the standard Compare and Merge Utility to both create and manage the Subject Area layers.

The key difference between this approach and the ‘Silo Design’ approach is in the way the Logical Data Model is defined. In both tools, we are forced to use a separate model file to contain the Enterprise Logical Data Model (which is a very good thing to maintain separation between them)

APPROACH 1 - SILO DESIGN

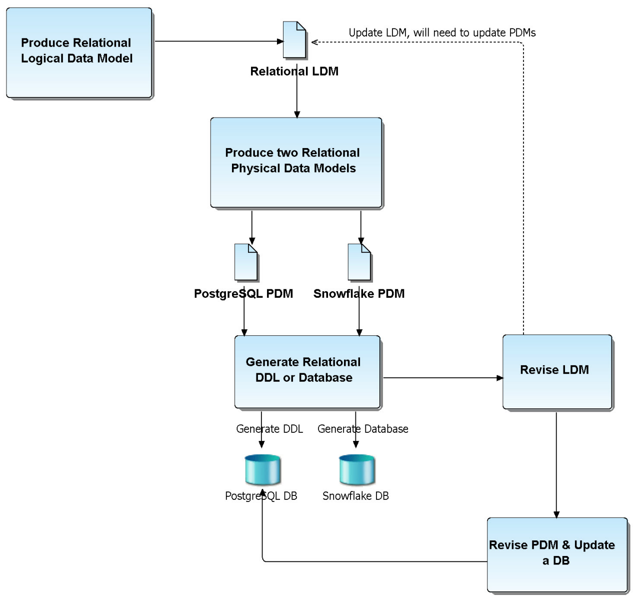
In the Silo Design approach, the Logical Data Model (LDM) may contain thoughts and ideas from other project teams or elsewhere in the organization, but the LDM is not linked in any way to anything apart from the project’s own Physical Data Models (PDMs).



STEPS

Here are the steps that I followed in both tools. I have provided a side-by-side comparison of the actions taken in each tool, so you can assess the two tools for yourself.

In a review such as this, the modeling and design process is necessarily simplified. There are project and modeling tasks that are not included in the comparison below, such as the setting of naming standards. However, the fact that these tasks have been excluded does not mean that they are not important.


Produce Relational Logical Data Model
Relational LDM
Update LDM, will need to update PDMs
Produce two Relational Physical Data Models
PostgreSQL PDM
Snowflake PDM
Generate Relational DDL or Database
Revise LDM
Generate DDL
Generate Database
PostgreSQL DB
Snowflake DB
Revise PDM & Update a DB

Create the Logical Data Model

ER/Studio Actions	erwin DM Actions
Create a new Diagram[1], which will automatically include an empty Logical Data Model	Create a ‘combined’[2] Logical / Physical Model
Add the required data model objects	Add the required data model objects

Observations

Both tools do an adequate job of creating the basic building blocks for a Logical Data Model.

Generate First Physical Data Model

ER/Studio Actions	erwin DM Actions
Use the “Generate Physical Data Model” wizard to generate a PostgreSQL PDM in the same file as the LDM.	The Physical View is automatic – a different view of the same objects shown in the Logical View.
In the Physical Data Model, use the denormalization techniques to roll the *Course* table down into the two child tables. Check the ‘Where Used’	In the Logical View, use the Supertype-Subtype Rolldown transformation to roll the *Course* table down into the two child tables. Check the ‘Where Used’

Observations

Both tools allow me to mark entities, attributes, and relationships as Logical-only, preventing objects from being included in the PDM if they are not wanted there. Both tools also allow me to mark PDM tables, columns, and relationships as Physical-only.

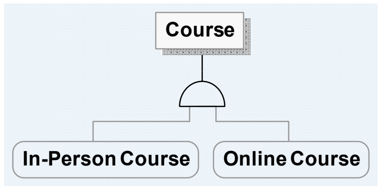
erwin DM

☒ ↓ Every action you take to denormalize a Physical View (such as rolling up a relationship in the Physical View) is mirrored in the Logical View; erwin DM does not maintain the separation of the Logical and Physical views of data. This is not unique to the ‘combined’ model either (as I refer to later in the review). Once you have closed the modeling session, erwin DM no longer ‘knows’ the original LDM structure. If you want to collapse a level in your Supertype-subtype hierarchy, there some limitations you need to be aware of:

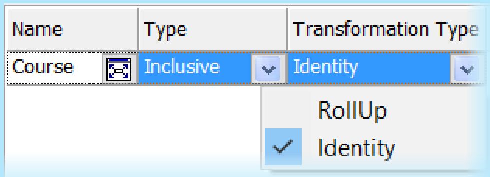
[1] The word “Diagram” has two meanings in ER/Studio. It can refer to an Entity-Relationship Diagram or (more often) it refers to a file that contains one or more data models. In ER/Studio each submodel can only contain one diagram, so a submodel is synonymous with an Entity-Relationship Diagram
[2] In erwin DM, a ‘Logical/Physical’ model is a single model with two views of the objects it contains – the label I prefer to use is ‘combined’

1. In the Physical View, you can only select one pair of tables for a roll-up or roll-down
2. After you have rolled-up or rolled-down, one of the tables is removed – this prevents you from rolling-down a parent table into more than one child table

erwin DM provides several different types of transformation for Supertype-Subtypes, some via the ribbon and some via the properties for a Subtype symbol. The documentation does not explain the differences clearly, so here is my interpretation of the options available.



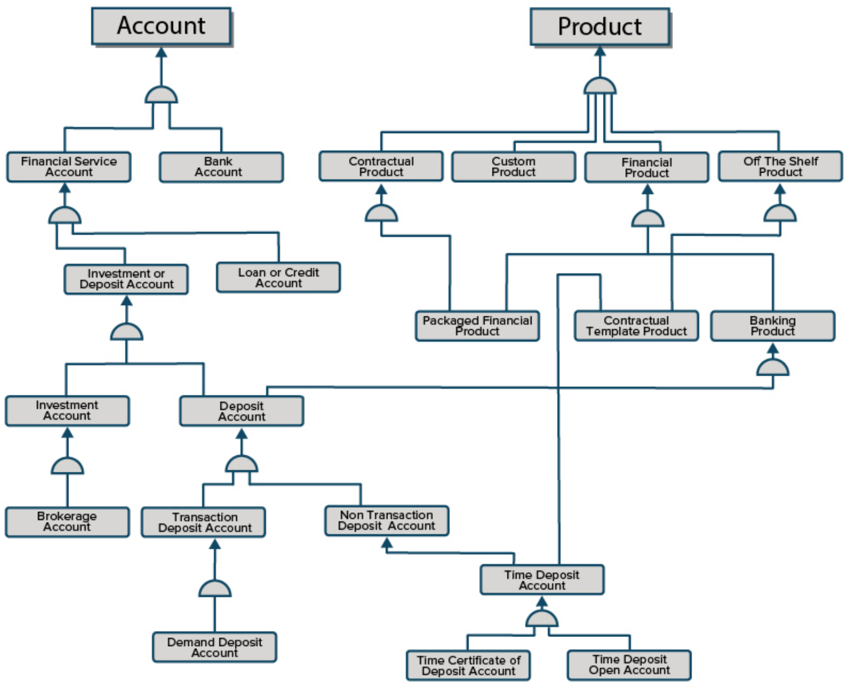
There are two different sets of actions you can take in the Logical View, depending on whether you have enabled "Supertype-Subtype Transformations" in the general Options (on the Tools ribbon).

Option is enabled	The available transformations are driven from the Logical View but only change the Physical View. In the properties for the Subtype symbol, you can change the default Transformation Type, from 'Identity' to 'Rollup' – 'Rolldown' should be available soon. The 'Identity' transformation converts the subtype links into one-to-one relationships in the Physical View; the 'Rollup' transformation ensures that the Subtype tables do not exist – they are all merged into the Supertype table. **This change cannot be undone once you have saved the model.** *Transformations* are not available on the *Actions* ribbon for Subtype symbols.  Name / Type / Transformation Type: Course / Inclusive / Identity; RollUp, ✓ Identity
Option is NOT enabled	*Transformations* are available on the *Actions* ribbon for Subtype symbols. These allow you to transform the Supertype-subtype set in one of three ways **in the Logical View** – Rollup, Rolldown, Resolve to Identity (replace Subtype links with relationships). In the Combined model, I found that sometimes no changes were actually made by these transformations.

¹ *For more information, see https://metadatamatters.com/fib-dm-in-er-studio/*

☒ Rolling-up or rolling-down a Supertype-Subtype hierarchy applies the action to every entity in that level in the hierarchy – you cannot be selective. In my sample model, for example, I cannot choose to roll-up Online Course into Course but leave In-Person Course untouched.

These limitations in erwin DM are amplified if your model has a lot of subtyping, such as the FIB-DM[1] model – see the diagram on the right for a small fragment of that model.



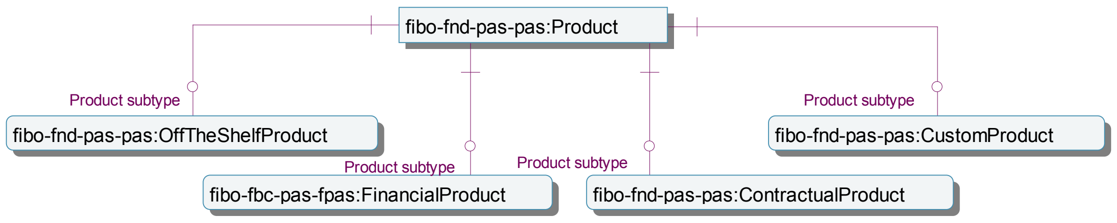
Account
Product
Financial Service Account
Bank Account
Contractual Product
Custom Product
Financial Product
Off The Shelf Product
Investment or Deposit Account
Loan or Credit Account
Packaged Financial Product
Contractual Template Product
Banking Product
Investment Account
Deposit Account
Brokerage Account
Transaction Deposit Account
Non Transaction Deposit Account
Time Deposit Account
Demand Deposit Account
Time Certificate of Deposit Account
Time Deposit Open Account

In the 'Where Used' dialogues, all entities/tables, attributes/columns, and relationships are linked to their equivalent in the other view (they are the same objects wearing different hats), unless marked as Logical-only or Physical-only.

ER/Studio

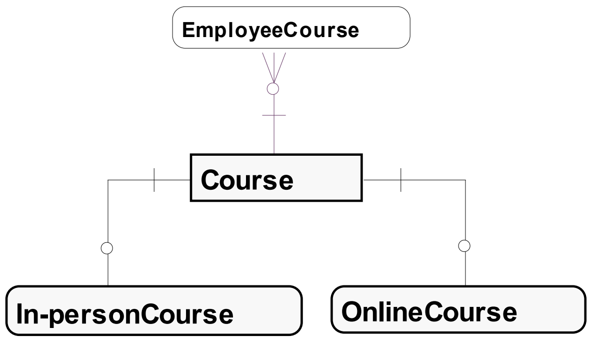
☑ All denormalization transformations are carried out in the Physical Data Model – the LDM is not affected.

☑ It is possible to roll-up or roll-down more than one PDM relationship at a time – I could, for example, roll-down fibo-fnd-pas-pas:Product into three of the subtypes in a single operation, leaving fibo-fnd-pas-pas:CustomProduct untouched.



fibo-fnd-pas-pas:Product
Product subtype
fibo-fnd-pas-pas:OffTheShelfProduct
Product subtype
fibo-fnd-pas-pas:CustomProduct
Product subtype
fibo-fbc-pas-fpas:FinancialProduct
Product subtype
fibo-fnd-pas-pas:ContractualProduct

I found that in ER/Studio – before I could use the roll-up or roll-down on the Course table I had to deal with the link to the table EmployeeCourse, which was automatically generated to resolve the many-to-many relationship with Employee.



There are two ways to do this:

1. Delete and recreate the relationship from Course to EmployeeCourse - ☒ this breaks the link back to the LDM
2. Resolve the many-to-many relationship in the LDM before you generate the PDM – this is my preferred action. Although ER/Studio provides the user a facility to normalize the model in the transformation. Idera, tell me as best practice, they encourage the designer to achieve Third Normal Form in the logical before generating the physical model(s).

In the 'Where Used' dialogues, all tables and columns are linked to the LDM objects, except for the tables derived from many-to-many relationships (Idera tell me they are linked behind the scenes where I cannot see them). If a table has been denormalized out of existence in the PDM, the original LDM entity is linked to the Transformation that removed the table.

☑ If required, I can also include Views in the LDM, which can then be generated into the PDM. Alternatively, Views in the LDM can be marked as Logical-only, perhaps being used to represent reporting or integration requirements.

Create the Database

ER/Studio Actions	**erwin DM Actions**
Use the *DDL Generation Wizard* to generate a database either using a direct connection or as one or more script files. I do not have the DBMS installed, so I generated a script file.	Use the *Forward Engineer Schema Generation* to generate a database either using a direct connection or as one or more script files. I do not have the DBMS installed, so I generated a script file.

Observations

The two SQL scripts generated are structured differently, but I'm able to reverse-engineer both of them into a third data model to compare the results – I could see that both tools generated acceptable SQL scripts (though I prefer the comments that were added to the ER/Studio script). Also ER/Studio can break the DDL script into smaller more manageable files.

Create the Second Physical Data Model

↓ There are a number of situations where a second PDM is needed:

- You need the same data structures to be available in more than one database (perhaps one is for reporting)
- You are migrating from one DBMS to another
- Integration projects where you need to understand semantic matches between data
- Master Data Management and Data Governance projects, again trying to understand where key data is deployed

ER/Studio Actions	erwin DM Actions
Use the "Generate Physical Data Model" wizard to generate a Snowflake PDM in the same file as the LDM and the existing PDM. Amend Naming Standards if required Rename table *TeamMember* to *TeamMembership* Check the 'Where Used'	erwin DM cannot have more than one PDM in a file – the only available option is to derive a new PDM from the LDM. There will be no initial connection between the LDM and this new PDM, but a subsequent Compare operation will produce links between the objects.

Observations

☑ ER/Studio allows more than one PDM in the same file, and they can all be radically different from the LDM.

☒ The 'combined' model in erwin DM cannot support multiple PDMs in one file, insisting that the PDM and LDM are two views of the same set of objects.

Edit the Logical Data Model

ER/Studio Actions	erwin DM Actions
Remove the Logical-only flag from one Entity and two Attributes. Change the data type for a Domain	

Observations

In both tools, when changing the Data Type for a domain, it also changed the Data Type for every Attribute and/or Column linked to the domain. In both tools you can prevent this by telling the tool that you want to override the properties inherited from the domain. In ER/Studio, select the 'Override Bound Data' option; in erwin DM, select the 'Override' option. ER/Studio supports user defined data types as their own objects in the dictionary whereas erwin only supports UDD's in the physical properties of the Domain

Update the Physical Data Model from the LDM

ER/Studio Actions	erwin DM Actions
Use the Compare and Merge utility to update the first PDM from the LDM.	The first PDM has already been updated - the second PDM cannot be updated in the same way because it's in a separate file.

Update the Database from the PDM

ER/Studio Actions	erwin DM Actions
Use compare and merge to generate ALTER file (you can also directly update the database). The script file generated earlier can be used to represent the current state of the database.	erwin does not support reverse-engineering a PostgreSQL script to create a model - which is necessary for the comparison - so cannot generate ALTER script or even verify the differences.

HOW WELL DO THE TOOLS SUPPORT THIS APPROACH?

As I mentioned before, both tools can create and manage the objects needed to represent the Logical and Physical Views of data. In the approach being tested here, all our models lie in one file – so be careful what changes you do or do not save.

↓ Both tools automatically resolve many-to-many relationships in the PDM though I suggest, as best practice, that you resolve those relationships in the LDM, instead of letting either tool make the decisions for you.

↓ Both tools support the use of object naming standards, along with the ability to synchronize logical and physical object names, though neither tool is foolproof. I was able to break the link between logical and physical objects in both tools by renaming the two linked objects (such as an entity and table). Both tools allowed me to link them together again, and ER/Studio allowed me to prevent this happening by enforcing full synchronization of logical and physical names.

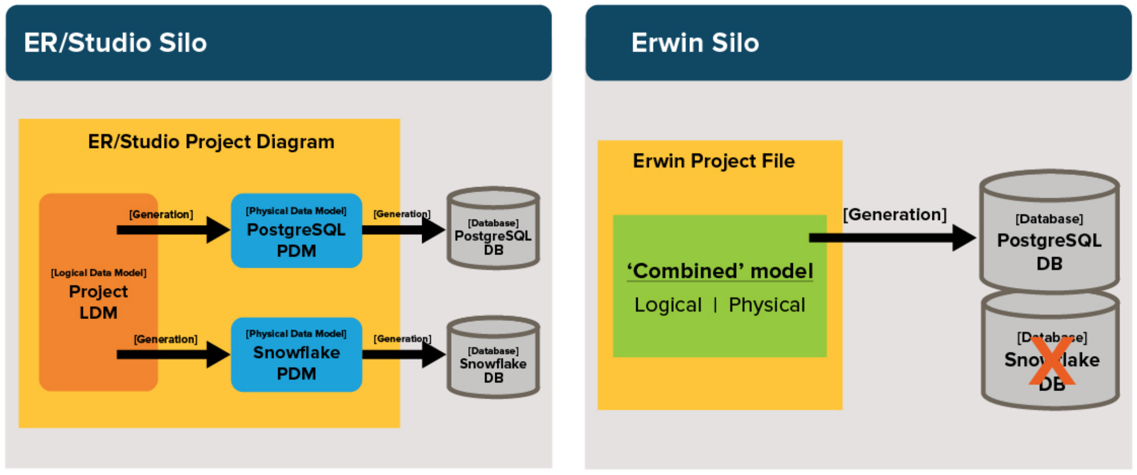
↓ ☒ erwin DM's 'combined' model has a very serious architectural flaw – it is impossible to separate the Logical and Physical views of data. If you have no subtyping in the Logical view and no denormalization in the Physical view this will not be a problem, but in my experience, that is not a realistic scenario.

ER/studio allows more than one PDM in the same file, and they can all be radically different from the LDM if needed. The 'combined' model in erwin DM cannot support multiple PDMs in one file, insisting that the PDM and LDM are two views of the same set of objects.

If I need to compare a PDM with a database that I'm not able to connect to, I will need to compare it with another PDM or with a script file that represents all or part of the database.

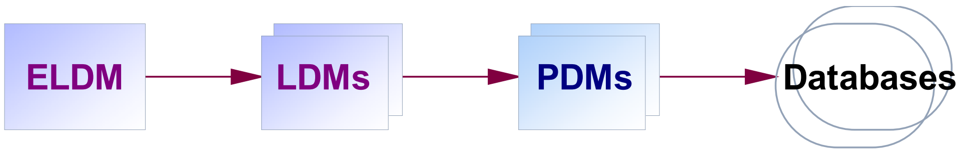
☒ In erwin DM my options are limited, as it cannot reverse-engineer SQL scripts for PostgreSQL and Snowflake. I must treat the SQL file as ODBC-standard SQL, which limits the objects and properties I can compare.

Here is a visual summary of the models created in each tool:



APPROACH 2 - ELDM-DRIVEN

In this approach, the Logical Data Model (LDM) contains more than just thoughts and ideas from other project teams or elsewhere in the organization, it contains data structures that have already been approved, and possibly represent data that is already available within the organization.



The steps followed are mostly the same as for the 'Silo Design' approach. I focus on the steps that are extra or different:

- the way the LDM is created and initially populated
- a feedback loop to the ELDM
- the way Physical Data Models are created in erwin DM
- the way Supertype/subtype clusters are handled in erwin DM.

STEPS

Creating the Logical Data Model

ER/Studio Actions	erwin DM Actions
Create a new Diagram – this will automatically create an LDM. Use the Compare and Merge utility to copy required objects across from the ELDM (this process can be repeated as requirements change). The scope of the merge could be defined using a submodel in the ELDM.	Open the ELDM Derive a new Logical Model using the Design Layers feature. The scope of the new model could be defined using a Subject Area in the ELDM.
Extend LDM where required	Extend LDM where required

Observations

Both tools allowed me to copy the items that I wanted in my new LDM, and create links back to the original model. In erwin DM the links were created automatically. In ER/Studio, I needed to save the mappings, where they are created as 'Universal Mappings'.

erwin DM

☑ The "Link Source Model" utility allows you to derive model objects from another model - this could be useful if you have already done some modeling work before you bring in content from the ELDM.

☑ The source models are listed in the Browser, so I can easily see that the LDM was derived from the ELDM.

☒ The documentation says that the source and target models are automatically linked when you derive a model, and that these links allow you to synchronize the two models at any time. However, this information is not visible through the user interface in the Source model. In the derived model, the model and objects have an entry in the history property that tells you the objects they were derived from. There is no apparent way to interrogate that history; perhaps to list the objects that do or do not originate from the ELDM. You can visualize the links by running *Sync with Model Source* in Design Layers.

☒ In my ELDM, I cannot see any information about the LDM that I derived from it. This information is visible in the third party Web Portal product though, and there is also a report that you can run from erwin DM that examines the links in the repository.

ER/Studio

☑ Once both models have been checked in to the Repository (Team Server), I can select the "Save As Universal Mappings" option. These mappings are recognized by the Compare and Merge utility, making it simple to match LDM objects to the equivalent ELDM objects, irrespective of what has happened to the object names. I can also create these Universal Mappings myself by choosing 'Save Mappings' when comparing two models or by using the Universal Mappings utility, which is useful where name matches don't exist, perhaps when mapping a siloed project model to the ELDM.

MERGING SELECTED PARTS OF LDM INTO ELDM

ER/Studio Actions	**erwin DM Actions**
Use the Compare and Merge utility to merge to and from the ELDM (this process can be repeated as needed).	Use "Sync with Source Model" to merge to and from the ELDM (this process can be repeated as needed). You can also use Complete Compare (see below for a note about this).

Observations

erwin DM

↓ I experimented with using both Complete Compare and Sync with Model Source to merge to and from the ELDM. Sync with Model Source was more successful at remembering the links between objects. The concept of linking objects in ERwin is to associate the object id's and store the pairs in the child model, save the model as an XML file and you can see these. This may have been user error but, I did find that the links would be forgotten if both linked objects have been renamed – matching them together in the comparison did sort the issue.

ER/Studio

↓ If I remember to save the mappings as Universal Mappings, ER/Studio correctly shows me the links.

GENERATING A PDM

ER/Studio Actions	**erwin DM Actions**
Use the "Generate Physical Data Model" wizard to generate a PDM in the same file.	Derive a new Physical Model using the Design Layers feature. The scope of the new model could be defined using a Subject Area in the LDM.

Observations

erwin DM

☒ ↓ Now that the models are in separate files, I cannot see the object names used in the ‘other’ model – when I look at an entity in the LDM I cannot see the table name(s). When I look at a table in the PDM I cannot see the entity name, except in the History property (which has a line telling me which object it was derived from).

Updating PDM with LDM changes

ER/Studio Actions	erwin DM Actions
Use the Compare and Merge utility to merge to and from the LDM (this process can be repeated as needed).	Use “Sync with Source Model” to merge to and from the LDM (this process can be repeated as needed). You can also use Complete Compare (see below for a note about this).

Observations

Both tools allowed me to create and update the items I wanted to update.

erwin DM

When I use Complete Compare to compare the LDM and PDM immediately after deriving the PDM, tables with multi-word names did not match up, neither do any of the columns. For example, the entity Team Member is not matched with the table TeamMember. To get around this, change the default comparison options, telling it not to compare the physical names – now everything lines up. I did not have the same issue using “Sync with Model Source” - it uses some metadata that the standard Complete Compare does not have access to.

DENORMALIZING THE PDM

ER/Studio Actions	erwin DM Actions
Experiment with rolling-down and rolling-up the subtyping structures and applying other normalization techniques – principally to test the separation of Logical and Physical Views.	

Observations

Both tools provide comprehensive denormalization features, though ER/Studio has some advantages from my perspective.

☑ ER/Studio can roll-down a table into several child tables in one operation

☒ erwin DM only processes two tables at a time – this makes it impossible to roll one parent table down into more than one child in erwin DM, unless you use super/subtype roll-downs in the LDM and merge the change into the PDM, which defeats the objective of having a normalized LDM.

☑ ER/Studio remembers the transformations that you carry out (linking them back to the LDM objects). In addition, ER/Studio can show you the state of the model before and after the transformation (via the Where Used tab for the PDM table and the affected LDM entities).

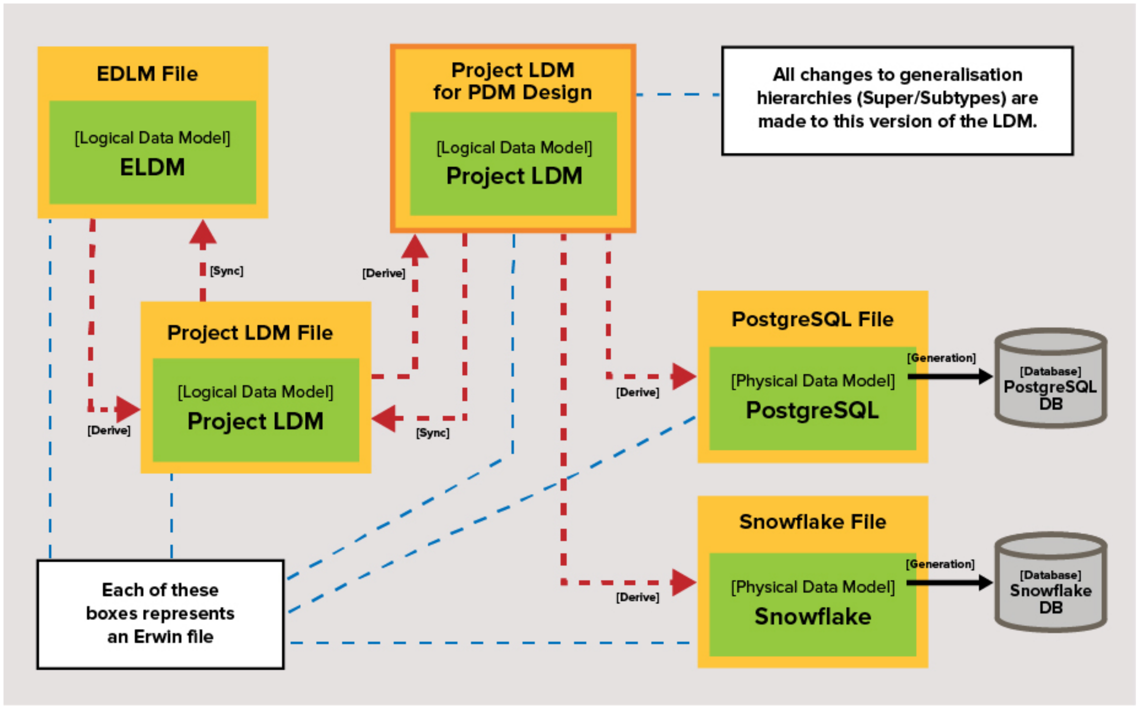
☒ In erwin DM, if I roll-up the OnlineCourse table into the Course table, a new line is added to History property in the Course table – that is the only record that I can see that recognizes that the transformation took place.

HOW WELL DO THE TOOLS SUPPORT THIS APPROACH?

Here is a visual summary of the models created in each tool:

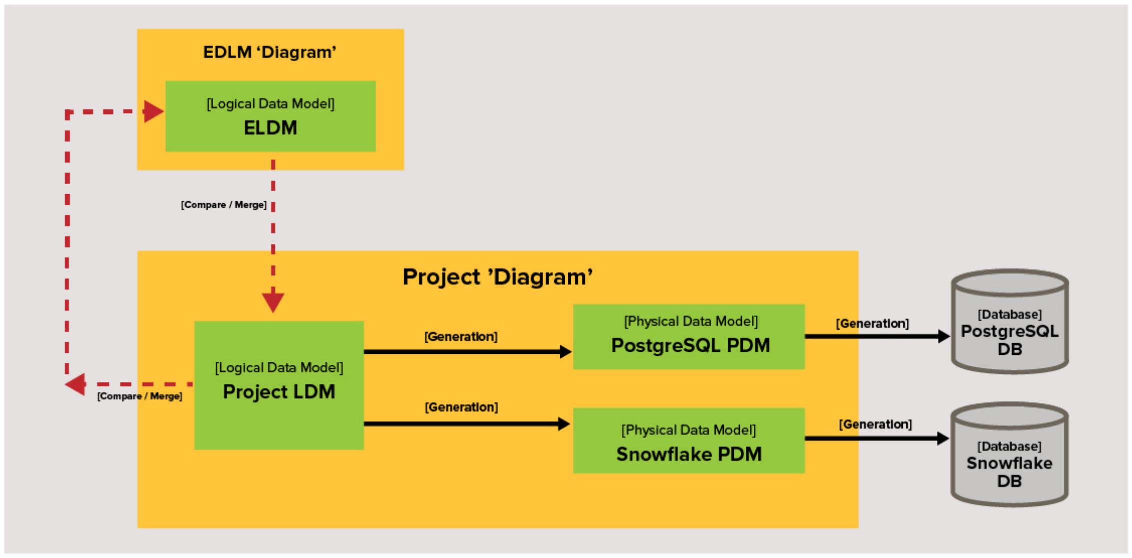
erwin DM

☒ Because erwin DM's forces me to carry out some of my denormalizations in the LDM, I have added a second project LDM for manipulating the many-to-many relationships and the Supertype/subtype hierarchies. If you need to resolve the Supertype/subtype issues differently for each database, the PDMs shown below will need to be 'combined' models.



As previously noted, the inter-model links can only really be visualized when using the Design layers feature to synchronize a model with a source, you would need to make your own notes about the links between the models. If you have the erwin Web portal, you can visualize the links there.

ER/Studio



☑ ER/Studio allows me to maintain the separation between the Logical and Physical views of data, so the number of models needed is reduced, as is the complexity of the tasks to carry out.

☑ ER/Studio Universal Mappings allow me to navigate nicely between the ELDM and the project models, with these relationships being visible in Data Architect and the web based Team Server product. Idera tell me that there will be new graphical visualizations of these relationships coming out in version 19, making it easier to navigate between models.

HOW DOES THIS AFFECT MY DBMS MIGRATION PROJECT?

Changing your preferred DBMS is a major task in any organization. If you are converting a database from PostgreSQL to Oracle or vice versa, you need an LDM to define the requirements for your new database. If you plan to merge two or more databases, a single LDM, that represents all business entities including common ones across multiple applications, is vital. If the LDM does not exist, you need to create it.

Here is an option for a model-driven migration approach. There are alternative approaches using a custom semantic model or an off the shelf ELDM. For this approach, you will need to:

- Reverse-engineer a PDM for the existing database(s)
- Derive the LDM for the database(s)
  - alter it if required (possibly based on the ELDM)
  - are you brave enough to sneak in new requirements at the same time?
- Create and refine the PDM for the new database
- Define the data mappings from the old PDM(s) to the new PDM

This process will involve at least three closely linked data models, along with the mappings that will be used to define how the data will flow from the current database(s) to the new database during the migration phase. Both tools allow you to define those flows via Visual Data Lineage in ER/Studio, and Data Movement Sources in erwin DM. Data Movement Sources are only available in the erwin PDM or a 'combined' model, whereas ER/Studio allows you to record Data Lineage in the LDM and PDM as well. This is very useful if, for example, you want to map multiple PDMs to a single LDM (such as the LDM for that Data Warehouse you are thinking of building).

I know from experience that lineage information is represented graphically in the ER/Studio user interface. There is no such view in erwin DM.

In erwin DM I would use the Design Layers feature to manage three or more separate model files, which I know it can do. I have real concerns about how I could keep track of the similarities and differences between the models, and the data mappings (as I have already stated). ER/Studio allows you to manage all those models in a single file, making it easier to keep track of similarities and differences, and the data mappings that are so necessary for a successful migration project.

In ER/Studio you can use the built-in Metadata import bridge to import ETL code. It will create several PDMs representing the source and target databases, in a single file, linking them together using Visual Data Lineage. erwin DM does not provide this automatic lineage creation capability, though it is available in the erwin Data Intelligence Suite which is a separate product to erwin DM.

CONCLUSION

SEPARATION OF THE LOGICAL AND PHYSICAL VIEWS OF DATA

Both tools are competent enough at the bread-and-butter modeling of databases – they will both create logical and physical data models, propagate changes from one model to the other in both directions, and create database scripts. Both will reverse-engineer databases. But they are not equal when it comes to maintaining the separation of the logical and physical views of data, a capability vital for maintaining good practice during the change process.

☑ ER/Studio maintains that separation.

☒ In the 'combined' (Logical/Physical) model, erwin DM forces you to break that separation. Using the Design Layers approach, erwin DM makes it too easy to break this separation by making it easier to deal with Supertype/subtype structures in the LDM instead of in the PDM.

Please remember that this is not a full evaluation, a bug-hunt, nor a click-by-click tool comparison.

TRACEABILITY THROUGH ALL THE LEVELS

Being able to manage the logical and physical modeling objects is not enough by itself – we need to be able to trace and visualize the links between the models. For example, the concept of a 'Team' could be represented by a series of linked objects in our models, such as the ELDM entity Team, several entities (probably also called Team) in project LDMs, and several tables in PDMs. I would like to be able to trace these links easily, preferably with a graphical view of the links. Unfortunately, neither tool provides a graphical view of the dependencies between objects, unlike some of their competitors (though both repositories make this possible).

In the absence of a graphical view of the links, I would like to be able to see the links in the user interface. Crucially, I want to be able to see the links from 'both ends', such as from the ELDM down to the LDM and from the LDM up to the ELDM.

☒ Without a repository, erwin DM users cannot follow the links from ELDM to LDM, to PDM.

☑ ER/Studio users can trace those links without a repository, using Universal Mappings.

ABOUT THE AUTHOR

George McGeachie is an independent information management practitioner, with 30+ years' experience creating and managing data (and other) models in many organizations. He encourages organizations to connect and utilize their metadata islands. He is a blogger, co-author of "Data Modeling Made Simple Using PowerDesigner" with Steve Hoberman, and author of several articles on TDAN.com. In case you're interested, he has hands-on experience of many metadata and modeling tools, including the Adaptive Repository, ASG Rochade, Bachmann GroundWorks, Bachmann Terrain & Terrain Map, erwin Data Modeler, ICL DDS, Idera ER/Studio Business Architect, Idera ER/Studio Data Architect, Intersolv Excelerator, MSP DataManager, Oracle Designer, Popkin System Architect, SAP PowerDesigner, and Visio Professional.

REFERENCES

[1] Hoberman, S. (2005). Data Modeling Made Simple, 1st Edition. New Jersey: Technics Publications.
[2] Hoberman, S. (2015). Data Model Scorecard. New Jersey: Technics Publications.
[3] International, D. (2017). Data Management Body of Knowledge. New Jersey: Technics Publications.
[4] Simsion, G. (2007). Data Modeling Theory and Practice. New Jersey: Technics Publications.