

Draft 2  
Data Modeling Tool Evaluation  
Larry Bednar  
September 18, 2003

## **Purpose**

This document summarizes the results of the author's evaluation of data modeling tools that can also be used to automatically generate database objects based on the contained data models.

## **Overview**

This section explains the sideboards of my evaluations, provides a high-level overview of my evaluation process, and provides an explanation of some of the commonly employed features of this type of software.

## **Background**

In my consulting practice, I find that I repeatedly create databases for clients that include many similarities. One of my normal practices is to create a relatively formal high-level entity relationship diagram to model the client's major business requirements, then to create a detailed entity-relationship model and finally something very much like a physical database model diagram as my understanding of the client's system needs increases.

I have worked with tools that can automatically translate such diagrams into database objects by generating SQL data definition language scripts, etc. However, until recently I had not been willing to spend the money to acquire such software for use in my current consulting practice. My prior experience was mostly based on use of Oracle's "CASE\*Designer" and "Designer 2000" software products, which are very comprehensive (really designed to support nearly all development phases of a data-centered development project), and rather expensive compared to the income of a relatively small time database consultant.

Instead, I've tended to use graphical design tools such as Microsoft Visio Professional 2002 to create graphical models, and then manually create the database objects needed to implement the model in the database management software selected for the client's use. Final database implementation is quite fast, as a well-honed graphical data model provides a very good map for manual implementation.

(Visio 2000 Enterprise, at least prior to Microsoft's purchase of Visio, contained capacities for reverse and forward engineering of data models and database objects using ODBC – My copy of Visio 2002 Pro does not provide such capacities. I believe that a purchase of Microsoft's high end versions of their "Net" products is now required to obtain these capacities from Microsoft products.)

However, it has recently become clear that a CASE tool's ability to automatically generate database objects from a data model would speed my development process, reduce costs for my clients, and keep me working at the conceptual level I like best for a greater portion of the time. So I decided to investigate the tools that I could determine to be available to see whether a suitable product was available.

My searches for published comparisons of these types of products and have not uncovered recent comparisons suitable for my use. So I decided to undertake an evaluation process myself, and decided to summarize my findings so that others might derive some benefit.

## **Common Software Features**

This section provides an explanation of some of the features commonly provided in these products. The generic terminology I've used to describe the feature may not correspond exactly to the label used in any one software product, but I've tried to apply labels that correspond reasonably well with the labels used in most of the products.

### **Entity-Relationship Diagrammer**

In the purest sense, an entity-relationship diagram (ERD) is a diagrammatic representation of the data subjects that must be tracked, with an indication of the inherent logical relationships between them. I myself usually think of an ERD as a form of business requirements documentation – the ERD documents the business's data requirements.

Each “entity” in an ERD represents “a thing about which information is tracked”. Examples might include: business account, person, an interaction between a staff member and a client, an order, a line item on an order, a document, a customer complaint, etc. Typically, an ERD uses a square or round-cornered rectangle to represent an entity. The name assigned to the entity is represented in or near the box.

Individual data items used to characterize an entity are called “attributes”.

Each “relationship” in an ERD represents a logical link between two entities. The relationships represented are typically pretty strong ones, typically concerned with conditions that must hold true to allow records to be created, or conditions that restrict the range of values that might be allowed within an attribute. This is not merely a case of indicating two attributes that might be used for a “join” operation, so readers who work heavily in FileMaker will have to readjust their understanding of what a relationship is to this more widely used definition.

Typical relationships might include these types of circumstances:

- The value assigned to an attribute in one entity must be a value already present in an attribute of a 2d entity
- An instance of one entity may not be created unless some instance of a 2d entity is unambiguously associated with it.

Theoretically, an ERD shouldn't really attempt any representation of details that are specific to implementation of the system in any particular hardware/software setting. So designations of data types to be used for attributes. However, it is pretty natural to take a “pure” ERD that contains no information pertaining to details of a specific DBMS implementation and begin adding details that slowly convert it into a very specific representation that really is not abstract or general any longer. At some point a purist would insist that at some point in this process, the model should no longer be called an ERD, but should instead be called a “database schema”, “a logical data model”, “physical data model”, etc. Most CASE tools using an ERD diagrammer allow entry of some details that probably really should not be in an ERD, and the point at which they begin making a distinction between the ERD and a “physical data model” varies.

### **Physical Data Model Diagrammer**

The “physical data model” represents the structures that will be implemented in the database. The terminology of the ERD is typically translated to related, but different concepts in the physical data model. An “entity” from the erd is typically represented as a “table” in the physical data model. A relationship is typically represented by foreign keys, and indexes, etc.

Although the ERD's logical representation of data relationships may separate two related entities, the developer may choose for valid practical reasons to implement both these entities within a single table in the targeted database. So the physical data model may present structures that seem similar to the ERD structures, but there will often be differences.

Many CASE tools provide an ability to indicate the estimated sizes of table rows (average size), the initial number of rows to be loaded into a table, the number of additional rows expected during some period of time, etc. Databases such as Oracle 8 often provide database administrators with storage options for which this information is useful.

## Domain Definitions

A "domain" is a definition of a data type with customized characteristics. The user typically is allowed to apply this cluster of characteristics to a column or attribute simply by selecting the name of the "domain" to be used. This is very much like selecting a data type. For instance the user might create a domain named "d\_def" as varchar(80), not null, etc. Then they can simply apply this domain to any attribute or column desired. This can save a lot of work during model building, and can simplify later changes. For instance, if the decision is made later that the "d\_def" domain must be varchar(35), this change can be made in all columns using that domain simply by changing the domain definition.

Most of the tools evaluated also provide reports that display a list of the attributes/columns that use each domain defined in the model.

## Reverse Engineering

"Reverse engineering" is the ability of a CASE tool to create a data model from an automated examination of a pre-existing database. Typically, an ERD and associated object definitions are created, with entity names corresponding to table names in the source database, entity attributes corresponding to table column names, etc.

All the evaluated tools that support reverse engineering create entities in an ERD to represent tables in the source database. Some also create objects representing "views" or "queries" defined in the source database. Methods for representing this type of object vary from product to product.

Most products allow the user to target a variety of DBMSs. Very common "source DBMS" options include: MS-Access, MS-SQL Server, Oracle, MySQL, ODBC. Because many groups in the non-profit work space use FileMaker, I'll mention here that non of the products I turned up for evaluation support a native targeting of "FileMaker".

## Model Synchronization

"Model synchronization" is the ability of a database to examine a database and update an existing data model to include any changes that have been made to the source database.

If you have a development environment where changes are frequently made in the DBMS during development, synchronization can be a powerful tool in ensuring that the current state of the system is clearly documented using ERDs and other data modeling tools.

## Forward Engineering

"Forward Engineering" is the ability to take the details described in the data model and generate scripts or program code that will create objects representing those features in a target DBMS. For instance, a "generate" function might produce a file containing VisualBasic code designed to create tables, columns, relationships and queries in a Microsoft Access database, or a file of SQL commands designed to create tables and columns in a MySQL database.

Some products in addition will create the database for the user.

Most products allow the user to target a variety of DBMSs. Very common “target DBMS” options include: MS-Access, MS-SQL Server, Oracle, MySQL, ODBC. Because many groups in the non-profit work space use FileMaker, I’ll mention here that non of the products I turned up for evaluation support a native targeting of “FileMaker”.

## **Version Control**

Some products provide the ability to archive versions of the data model and to restore these versions later.

## **Model Reports**

Most of the evaluated products provide the ability to generate text, HTML, XML, or RTF documents that outline the system details defined within the CASE tool.

## **Object Explorer**

Most tools provide some “explorer” type interface that facilitates the user’s navigation through and examination of details defined for entities, attributes, relationships, indexes, views, domains, etc.

## **To-Do Lists**

Several of the examined products provide the ability to record “to-dos” for various data model objects within the CASE tool itself.

## ***Common Software Features***

This section provides an explanation of some of the features commonly provided in these products. The generic terminology I’ve used to describe the feature may not correspond exactly to the label used in any one software product, but I’ve tried to apply labels that correspond reasonably well with the labels used in most of the products.

## **Entity-Relationship Diagrammer**

In the purest sense, an entity-relationship diagram (ERD) is a diagrammatic representation of the data subjects that must be tracked, with an indication of the inherent logical relationships between them. I myself usually think of an ERD as a form of business requirements documentation – the ERD documents the business’s data requirements.

Each “entity” in an ERD represents “a thing about which information is tracked”. Examples might include: business account, person, an interaction between a staff member and a client, an order, a line item on an order, a document, a customer complaint, etc. Typically, an ERD uses a square or round-cornered rectangle to represent an entity. The name assigned to the entity is represented in or near the box.

Individual data items used to characterize an entity are called “attributes”.

Each “relationship” in an ERD represents a logical link between two entities. The relationships represented are typically pretty strong ones, typically concerned with conditions that must hold true to allow records to be created, or conditions that restrict the range of values that might be allowed within an attribute. This is not merely a case of indicating two attributes that might be used for a “join” operation, so readers who work heavily in FileMaker will have to readjust their understanding of what a relationship is to this more widely used definition.

Typical relationships might include these types of circumstances:

- The value assigned to an attribute in one entity must be a value already present in an attribute of a 2d entity
- An instance of one entity may not be created unless some instance of a 2d entity is unambiguously associated with it.

Theoretically, an ERD shouldn't really attempt any representation of details that are specific to implementation of the system in any particular hardware/software setting. So designations of data types to be used for attributes. However, it is pretty natural to take a "pure" ERD that contains no information pertaining to details of a specific DBMS implementation and begin adding details that slowly convert it into a very specific representation that really is not abstract or general any longer. At some point a purist would insist that at some point in this process, the model should no longer be called an ERD, but should instead be called a "database schema", "a logical data model", "physical data model", etc. Most CASE tools using an ERD diagrammer allow entry of some details that probably really should not be in an ERD, and the point at which they begin making a distinction between the ERD and a "physical data model" varies.

## Physical Data Model Diagrammer

The "physical data model" represents the structures that will be implemented in the database. The terminology of the ERD is typically translated to related, but different concepts in the physical data model. An "entity" from the erd is typically represented as a "table" in the physical data model. A relationship is typically represented by foreign keys, and indexes, etc.

Although the ERD's logical representation of data relationships may separate two related entities, the developer may choose for valid practical reasons to implement both these entities within a single table in the targeted database. So the physical data model may present structures that seem similar to the ERD structures, but there will often be differences.

Many CASE tools provide an ability to indicate the estimated sizes of table rows (average size), the initial number of rows to be loaded into a table, the number of additional rows expected during some period of time, etc. Databases such as Oracle 8 often provide database administrators with storage options for which this information is useful.

## Submodels

A "submodel" is a representation of a portion of the complete data model. In cases where the data model is very complex, this representation may facilitate discussion with users, or may allow some areas of the model to be examined more clearly.

## Domain Definitions

A "domain" is a definition of a data type with customized characteristics. The user typically is allowed to apply this cluster of characteristics to a column or attribute simply by selecting the name of the "domain" to be used. This is very much like selecting a data type. For instance the user might create a domain named "d\_def" as varchar(80), not null, etc. Then they can simply apply this domain to any attribute or column desired. This can save a lot of work during model building, and can simplify later changes. For instance, if the decision is made later that the "d\_def" domain must be varchar(35), this change can be made in all columns using that domain simply by changing the domain definition.

Most of the tools evaluated also provide reports that display a list of the attributes/columns that use each domain defined in the model.

## Reverse Engineering

“Reverse engineering” is the ability of a CASE tool to create a data model from an automated examination of a pre-existing database. Typically, an ERD and associated object definitions are created, with entity names corresponding to table names in the source database, entity attributes corresponding to table column names, etc.

All the evaluated tools that support reverse engineering create entities in an ERD to represent tables in the source database. Some also create objects representing “views” or “queries” defined in the source database. Methods for representing this type of object vary from product to product.

Most products allow the user to target a variety of DBMSs. Very common “source DBMS” options include: MS-Access, MS-SQL Server, Oracle, MySQL, ODBC. Because many groups in the non-profit work space use FileMaker, I’ll mention here that non of the products I turned up for evaluation support a native targeting of “FileMaker”.

## Model Synchronization

“Model synchronization” is the ability of a database to examine a database and update an existing data model to include any changes that have been made to the source database.

If you have a development environment where changes are frequently made in the DBMS during development, synchronization can be a powerful tool in ensuring that the current state of the system is clearly documented using ERDs and other data modeling tools.

## Forward Engineering

“Forward Engineering” is the ability to take the details described in the data model and generate scripts or program code that will create objects representing those features in a target DBMS. For instance, a “generate” function might produce a file containing VisualBasic code designed to create tables, columns, relationships and queries in a Microsoft Access database, or a file of SQL commands designed to create tables and columns in a MySQL database.

Some products in addition will create the database for the user.

Most products allow the user to target a variety of DBMSs. Very common “target DBMS” options include: MS-Access, MS-SQL Server, Oracle, MySQL, ODBC. Because many groups in the non-profit work space use FileMaker, I’ll mention here that non of the products I turned up for evaluation support a native targeting of “FileMaker”.

## Version Control

Some products provide the ability to archive versions of the data model and to restore these versions later.

## Model Reports

Most of the evaluated products provide the ability to generate text, HTML, XML, or RTF documents that outline the system details defined within the CASE tool.

## Object Explorer

Most tools provide some “explorer” type interface that facilitates the user’s navigation through and examination of details defined for entities, attributes, relationships, indexes, views, domains, etc.

## **To-Do Lists**

Several of the examined products provide the ability to record “to-dos” for various data model objects within the CASE tool itself.

## **Evaluation Criteria**

My evaluation is mostly subjective, but there were some overarching concerns that I tried to always keep in mind:

1. I wanted the ability to use graphical data modeling tools to construct and examine the data model.
2. I wanted to be able to generate “object creation” scripts directly from the CASE tool. I viewed it as a bonus if the tool would also let me directly create a database and associated objects, but even in this case, I wanted to be able to store a command file for examination and reuse.
3. I hoped to find a tool that would allow me to generate PDF representations of data models for distribution to clients for development review or as a part of system documentation delivered to them at completion of my work.
4. I wanted a tool that would generate database objects in both Microsoft Access and in MySQL, since I expect to work in both these platforms frequently in the near future.
5. I wanted a tool that would allow reverse engineering – the automated creation of a data model from the software’s examination of an existing database.
6. Ideally, I wanted a tool that would be capable of examining a modified version of a database and updating pre-existing data models to reflect manual changes that have been made in that database.
7. I did not want to pay for software for evaluation purposes. I only examined software that I could obtain in free trial versions that were more or less “full featured”.
8. My own limit for expense is around \$1500. I did not attempt an evaluation of more expensive products, even if I was aware of them.

I used a populated copy of my “time tracking database” (MS-Access), available on my web site’s download page as a source database for reverse engineering tests.

Forward engineering was tested by attempting to create a Microsoft Access database from the data model resulting from reverse engineering of my “time-tracking database”. Additional tests for creation of MySQL database objects were performed simply by generating the “create” scripts and reviewing those scripts manually for obvious errors. Since much of MySQL DDL is the same as or similar to standard SQL DDL, I did not feel that these scripts were likely to contain incorrect statements, etc.

All evaluations were done on an Intel Pentium III personal computer running MS-Windows 2000 Professional. I installed binary versions of all products designed specifically for operation on Windows operating systems.

## **Disclaimer**

I was attempting to gain a fast overview of these products. I do not doubt there are options and features of value in many or all of them that I did not have time to evaluate. In fact, it is possible that features I feel are lacking are instead available through option settings that were not obvious during my very quick evaluation.

In addition, I attempted no “stress testing” of any kind. I did not attempt to use the products with highly complex models, specifying unusual data types and then attempting to generate scripts for databases that do not support those data types, etc. I did not attempt connections to model repositories over local area network or internet connections. And I did not attempt to test security.

I encourage the reader to keep in mind that the subjective evaluations provided here are from my own perspective. I’ve used some comprehensive CASE tools in the past (Oracle’s CASE products), and I use many of the typically tools (diagrams, etc) as a routine in my own projects. I have attempted to evaluate ease of use for the general reader, but of course these conclusions might be influenced by my own background and experience.

## **Software Evaluations**

This section contains my summary of individual product evaluations.

### ***Products Evaluated***

I did searches on Google and Yahoo to identify as many products as possible that should be evaluated. Interestingly enough, searches using all the keywords I could think of (“data model”, “entity-relationship”, etc.) turned up only Oracle Designer and ERWin Data Modeler. Or at least nothing else came up high on the listings returned by my searches. A later search through software listed on the MySQL web site as working with MySQL provided more candidates. Web sites devoted to database design/administration provided links to additional products.

The following products were examined:

1. CASE Studio (CharonWare)
2. DB Designer 4 (FabForce)
3. DDS-Lite (Chili Source)
4. Dezipn (Datanamic)
5. ERWin Data Modeler (Computer Associates)

### ***Installation***

Installation for all products was relatively smooth. In one case, I had to abort an install that I had started as a non-administrative user of my computer. You might make a note to install as an administrator if you decide to make use of any of these products.

### ***Evaluation – CASE Studio 2, V2.13***

At the time I downloaded, CASE Studio could be purchased for about \$330, in U.S currency. Additional licenses were available at around \$100 each. It seems the product is produced in the Czech Republic, and there are a *few* places in the on-line help where a sensitive reader will guess that the text has been translated from another language. However, the indications of this translation are subtle - typically minor grammatical errors that do not hinder the American reader from correctly understanding the author’s meaning.

I found the software easy to use, but I encourage the reader to remember that I’ve used some pretty complex CASE tools previously (Oracle’s Designer provides tools for modeling business processes, business entities, database objects, and some other types of software system features) and I use general data modeling techniques on almost every project.

CASE Studio provides the ability to specify a LOT about entities and attributes, indexing, relationships, referential integrity (as it should).

CASE Studio also provides the ability to work with dataflow diagrams, a tool that I've often found useful. However, I've not found many other developers who make frequent use of data flow diagrams.

The definition of "submodels" is supported.

A version manager is also provided.

User accounts can be set up within CASE Studio.

CASE Studio provides "to-do" lists within each area of the model, so that notes on development needs can be entered directly into the model.

CASE Studio allows the definition of "user data types" that represent customized data types. The use of domains is also supported. However, not all target DBMSs support domains.

CASE Studio also supports definitions of user accounts and roles for users of the software. This might be an advantage where several persons are working on the same project.

## Reverse Engineering Results

Reverse engineering of the "time-tracking database" resulted in all tables being correctly represented in CASE Studio's entity relationship diagrammer. All descriptions entered for columns and tables in the source database were correctly represented in the data model created by CASE Studio. Indexes were represented correctly, including unique indexes on primary keys and indexes placed on foreign keys. However, the relationships defined in the source database were *not* represented in CASE Studio. CASE Studio's on-line help seemed to indicate that relationships *should* have been correctly created, but they were not.

CASE Studio imports MS-Access queries as "text objects" which are named using the query name. Each of these objects contains the SQL code that defines the query. It appears that even SQL statements used to define data sources for combo and list box controls on MS-Access forms are imported.

## Forward engineering

A little bit of manual effort is required of the user. To generate objects in an MS-Access database, the user must create the new database, create a new code module in that database, open that module and copy in the contents of a DAO 3.6 procedure that CASE Studio creates for the user. A user who's done enough Access development to be comfortable opening the VisualBasic code editor should have little difficulty. The scripts I generated in 2-3 iterative tests all seemed to run just fine. The tables represented in CASE Studio's entity-relationship diagrammer were correctly created in my new MS-Access database. Queries were also correctly defined in the new Access database.

Since I'm somewhat obsessive about documentation, I was also pleased to see that all table and column descriptions were correctly copied into the newly created database. Most of the other products I tested did not perform this nice step, at least not with the options my fast evaluation led me to use. It is of course possible that I would find options for this feature given more time with the other products.

MySQL is explicitly supported.

## Overall Evaluation

I feel that this software is a great value. It would seem to be a great choice for anyone really interested in streamlining their data development process and formalizing their development process somewhat. All the basics seem to be covered in a straightforward, accessible fashion.

### ***Evaluation – DB Designer 4***

DB Designer is open source software licensed under the Gnu Public License. There is no charge for use of this product.

DB Designer is designed specifically for use with MySQL. As such, it really doesn't fit the parameters for my own use, but I felt that reviewing this software would provide utility to some readers of this document.

DB Designer's on-line documentation indicates the capacity to synchronize models with databases based on them. Changes made in the database therefore can be automatically reflected in the data model.

DB Designer's user interface frequently feels slow or unresponsive. I frequently found myself click mouse buttons repeatedly in situations where it usually seemed that a single click was adequate. It also seemed that the pop-up help text that appears when the mouse cursor is passed over tool icons was not consistently responsive. Not really a major issue, but a minor irritation. In addition, I found the tool icons to be somehow slightly more difficult to understand than in most of the tools I evaluated. The organization of the tools also seems just a little uncomfortable, although it is difficult to describe why.

## Reverse Engineering

DB Designer can reverse engineer a data model from connections to MySQL, SQLite, Oracle, MS SQL Server, or ODBC databases. As of 17Sep2003, I did not have time to check reverse engineering of a MySQL database.

## Forward Engineering

DB Designer offers an option to order CREATE TABLE statements in the generated file in order of table linking by foreign key references.

The output script for database object creation all looked to be a proper implementation of SQL for MySQL. I like my hand-written SQL formatted a bit more heavily, but the formatting applied was quite acceptable: indentation was good, use of whitespace was good and SQL keywords were uppercase, with object names in lowercase.

DB Designer also can create OPTIMIZE and REPAIR scripts for selected tables or for all tables in the database. Since running optimization might be a standard part of a database maintenance routine for MySQL, this is of some value, however, the commands contained in these scripts are quite simple ("REPAIR table\_name", etc.), so few DBAs will feel that this capacity is of critical importance – it's just a minor convenience.

## Overall Evaluation

DB Designer provides the most needed functions, and would probably be a nice toolset addition for anyone who worked primarily in MySQL databases. Some of the apparent oddities in interface behavior are minor irritations, but don't seem to really hinder use of the core functions delivered in this product. It doesn't really have the level of polish of most of the software packages I evaluated, but then "free is a very good price". (For the money, it is an excellent value!)

## **Evaluation – DeZign for Databases v3.0.2**

At the time I downloaded by 30 evaluation copy, purchase of a downloaded copy of this software was a \$229 expense.

DeZign does support use of domains within it's data models. On-line help claims that DeZign will correctly translate domains defined for modeling use if the target DBMS does not formally support domains. Domains do not show up as if they are just another generic data type that might be assigned to an attribute, but instead are set by the user in an entirely different property of an attribute – this makes it easy for the user to distinguish their application of a user-defined domain from the use of a standard data type available in the target database. This distinction might be useful – in CASE Studio and DDS-Lite, I made this distinction clear by using a distinctive naming approach to domains I defined for use. While this “naming convention” approach is easy enough, DeZign's approach would prevent any confusion whatsoever, no matter whether the user forgets to use their agreed naming convention for domain names.

DeZign feels more strongly oriented to the “object explorer” used to display a “tree” of all model objects than CASE Studio, or DDS-Lite. In some subtle fashion, it seems clear to the user that all the diagrams etc are really supporting tools for the object explorer. In some of the other tools here, the diagrammers and other tools seem to play a more prominent role from the user's perspective.

### **Reverse Engineering**

DeZign has separate “reverse engineering” scripts for MySQL, Access, SQL script, and Paradox/dBase information sources. The tools that perform these functions are not provided by default with the purchased or evaluation copies of DeZign. All these importers were available in a single purchase for \$129. The importers can be downloaded in evaluation versions.

The “ImportER Access” tool requires the source database to make use of ADO objects. Options to show system objects must be set, and at least some of the system objects must be set to “read” access.

The evaluation version of ImportER Access will not import more than 8 attributes per table, so my complete database was not recreated in DeZign.

DeZign supports the storage of multiple versions of a project file.

DeZign offers separate “to-do” text areas within the “properties” collection of most objects in the data model.

DeZign did not import table or column descriptions from the source MS-Access database.

DeZign did not import MS-Access queries in any fashion that I could detect.

### **Forward Engineering**

DeZign created the new “targeted” MS-Access database as a part of the “generate” procedure. I received a message that seemed to indicate that one of the generation options that I'd selected was invalid for the generation I'd requested. This window appeared 5 times, which is the same as the number of entities in my data model, the number of “autoincrement” columns defined in the whole model, etc. Despite the warnings, it appeared that the results of the “generation” action were correctly constructed tables

Indexes represented in the data model were correctly defined within the target database.

Descriptions entered for entities and attributes in DeZign were not represented in the newly created MS-Access database.

The DDL script created for generation of MySQL database objects looked correct, although for one of the 5 tables, I expected a “UNIQUE KEY” clause for the primary key in a CREATE TABLE statement and none was produced.

DeZign generated a warning message when I generated a database without using a defined domain. I actually appreciated this level of checking...

## **Overall Evaluation**

A good tool, and I think another good value in terms of the time it might save, and the quality of product that would be facilitated if this software was used to full advantage as an alternative to manual methods.

I myself do not like the provision of “reverse engineering” capacities as separate tools, although I can appreciate that there might be nice development advantages to this approach and it really caused little difficulty. I find it to be at least a minor convenience when these capacities are more completely integrated with the main product.

### ***Evaluation – DDS Lite V2.11.0***

At the time I downloaded this product, copies could be purchased through download for less than \$100. Although Chili Software seems to be located in Singapore, all the on-line help seemed to employ very good American English – I would never have guessed that the software wasn’t of United States origin.

In general, most data modeling products use diagramming conventions in their “entity-relationship” diagrams that are closer to what is traditionally called a “logical data model”. In the “logical data model” conventions that are typical, many data features that are perhaps irrelevant to a pure “entity-relationship” model are often represented. These include representations of indexes, foreign keys and column data type designations. Theoretically, it could be argued that such details really don’t belong in the “entity-relationship” diagram, which is supposed to indicate abstract information structure rather than software implementation details.

The diagramming conventions used for entity-relationship diagrams in DDS-Lite are a little more “pure” than those used by most other products. The entity relationship diagram here really doesn’t display many attributes like assigned data types, indexes, etc. DDS-Lite takes a more “theoretical” line on the “entity-relationship” diagram, and provides a “Data Structure Editor” that presents the kind of details seen in the entity-relationship diagrams of most of the other products I evaluated.

This separation of data modeling into relatively distinct “theoretical” and “physical” components really isn’t hard to grasp. It probably didn’t slow me down more than 5-10 minutes. Actually, I found myself thinking that the use of a more “theoretical” approach to entity-relationship diagramming probably is beneficial if it forces the user into a more thorough high-level consideration of their situation before they start drilling into specific DBMS-related implementation details...

DDS-Lite does provide a “domain templates editor”, which seems to be designed to support the user’s definition of column attributes that will be used repeatedly throughout the implemented system.

During my effort to understand how domains would be used, I found that the on-line help system didn’t seem to properly link from the index entry for “domains” to an appropriate help page. And I also could not seem to find any other information on the subject. The on-line help didn’t really help much in this topic – at least not anything I could find.

I also found that the on-line help seemed to be dead wrong about one other item – the definition of properties for a relationship. Following the instructions provided for this task in the on-line help did

not provide the screens described. For some of the relationship attributes that were supposed to be available (according to on-line help), I was able to find no way of accessing or displaying them. My sense was that the approach to diagramming relationships had changed, but the on-line help did not reflect those changes.

Once a user defines a “domain” in a schema, that domain shows up as one of the possible “datatype” selections the user can apply to a within the data structure editor.

DDS-Lite also provides an “SQL keywords” editor. This tool is used to parse object names for problematic uses of SQL keywords, and to cause correct color-presentation of SQL keywords during code review. This is a nice feature. I’ve actually worked with Oracle databases that were implemented with columns named “date” – it can be difficult to figure out what is wrong in such a case, since any SELECT query you write that includes the use of the word “date” where SQL expects a column name yields an error.

DDS-Lite does not seem to provide any way to represent (or generate) “views”.

Tables indicating the translation of internal data types into data types for a target DBMS are available for ready review by the user.

I did not find support is provided for data volume information.

## Reverse Engineering

This product does not support the creation of an internal data model based on an existing database. Chili Software indicates that this capacity will be provided in the “DDS-Full” product that entered testing during August 2003. I was not able to get a copy of that more full-featured product, and probably would have chosen not to evaluate a “test” release of that software anyway. However, I expected the “full” copy to be quite similar to the “Lite” version, so proceeded with the evaluation of the existing DDS-Lite product...

## Forward Engineering

Many of the products I evaluated require the user to point the “object generation” towards an existing database. For MS-Access “targets”, DDS-Lite actually creates an Access database. In fact, DDS-Lite will actually overwrite an existing Access database during this creation process. I found this handy, but I still want to be able to save the script containing the relevant commands, which DDS-Lite also allowed. The database objects in the new database are created using SQL statements that are stored internally in DDS-Lite. The user can review either the “create”, “drop”, or “load” scripts that are created from within DDS-Lite, with simple syntax highlighting provided. You can generate scripts targeting several DBMSs simultaneously – the scripts are clearly named and placed in a different OS folder for each target DBMS. (I liked this approach to “generation” – I got everything I needed to evaluate at one fell swoop.) If you want to immediately load data into these database objects, the created “load” script would give you a nice start.

DDS-Lite did correctly create tables, relationships, and primary key indexes. However, column and table descriptions entered into DDS-Lite were *not* represented in the resulting MS-Access database.

DDS-Lite explicitly supports MySQL.

## Overall Evaluation

DDS-Lite seems to be a pretty good CASE tool, containing most of the features that I would expect. The biggest lack seems to be the apparent lack of support for views. In many cases, this would be of little concern. For instance, MySQL V4 and earlier do not support views, so a designer working

primarily in MySQL wouldn't find this "omission" relevant at all. Considering the price, it seems like a great value. The lack of support for data volume estimates may be of concern to those working in DBMSs that require such specification by the user (Oracle 8, etc.).

### ***Evaluation - ERWin Data Modeler***

ERWin Data Modeler was by far the most expensive of the products I reviewed. A single-user license at the time I evaluated these products was over \$1200.

However, for the right user ERWin is probably well worth the additional money. The internal approach to model building is much more technical and rigorous. Features such as tools to construct naming rules, abbreviation glossaries, and specify uses of this information that may differ from entities to tables to other objects are provided. ERWin would definitely take more time to learn thoroughly, but for someone frequently involved in database design overall benefit could be realized in just a small number of projects.

ERWin provides the user a lot of detail in detailing data volume, which would be important for those targeting really serious DBMS climates such as Oracle, DB2, etc.

ERWin also supports the establishment and use of formal "domains", which allows the user to define column characteristics that will be used repeatedly in their development and apply those specifications repeatedly by simply referencing the label for the "domain" they want to use. In a thoughtful design process, this can save a lot of time.

ERWin is also the only one of the products reviewed that provided features to "update" an existing data model after examining changes that have been made to a database developed from that model.

ERWin supported the specification and generation of "views". See the "reverse engineering" section below for some interesting details.

ERWin provides a notable distinction between "logical" and "physical" models. "Logical" models employ entities, "physical" models represent "tables" and "views".

ERWin also supports the categorization of entities into different "subject areas", which should provide a very nice capacity to break a very large, complex model into smaller "topic areas" that are easier to work with, easier to discuss with system users, etc.

### **Reverse Engineering**

ERWin did a very thorough job of representing the source database within its internal data model.

All table and column descriptions were represented correctly within ERWin's internal model.

ERWin representations of MS-Access queries were represented as "views" in the "physical" part of ERWin's data model. In fact, the relationship of these views to tables was also represented in the "physical data model" diagrams that were automatically produced. ERWin in addition seemed to reproduce SQL code written into the "data source" property of MS-Access forms and controls.

### **Forward Engineering**

ERWin created a VisualBasic file containing commands to create database objects in the target database. I believe this script used DAO 3.6 objects.

ERWin seems to require that the target database is already created by the user, although it seems to also expect that the database is empty. I got an error message and the "generation" process was stopped (thankfully) in one case where I accidentally specified a "target" database that already had copies of several of the tables ERWin was being asked to create.

All views represented in ERWin's physical data model seemed to be correctly represented in the target Access database.

All tables and indexes were also correctly recreated in the target MS-Access database. However, column and table descriptions were not represented during my first effort. A short investigation of the "generation options" available led to the understanding that I had to specify a non-default option to get that result. After enabling the appropriate option, all Access columns and tables included the appropriate descriptions.

The closest ERWin would come to direct support of MySQL was a "Target SQL DBMS" setting of "Generic ODBC". Nonetheless, I would expect this setting to work pretty well for a user capable of manually parsing SQL code to ensure adherence to MySQL SQL syntax.

## **Overall Evaluation**

ERWin is a great tool for a highly technical user who really understands database theory pretty well. Such a user, if they performed database design tasks regularly would welcome it's wealth of features and options, and it's rigorous, formal approach. For a user with less knowledge of database theory, or less regular need to use the software, ERWin would probably feel like overkill, and much of the power provided may actually interfere with that user's application of the software.