



Feature Matrix

| GENERAL DATA MODELING | XE | DATA ARCHITECT |
|--|----|----------------|
| Standard Modeling Support | | |
| Automatic propagation of a foreign key from parent to child entities in a logical model | x | x |
| Automatic propagation of a foreign key from parent to child entities in a physical model | x | x |
| Automatic Removal of foreign key upon Relationship Deletion | x | x |
| Automatic propagation of PK column data type changes | x | x |
| Supports the IDEF 1X notation for logical modeling | x | x |
| Supports the IDEF 1X notation for physical modeling | x | x |
| Supports the Information Engineering (IE or Crows Feet) Notation for logical modeling | x | x |
| Supports the Information Engineering (IE or Crows Feet) Notation for physical modeling | x | x |
| Supports a unique/specific Logical modeling environment | x | x |
| Supports multiple physical models and easily derives individual physical models from logical model | x | x |
| Supports dimensional modeling with ability to model star and snowflake schemas | x | x |
| Compare and merge changes between logical and physical models | x | x |
| Compare and merge two separate logical models | x | x |
| Compare and merge changes between physical models of the same DBMS | x | x |
| Performs common denormalization techniques like roll up, roll down, horizontal and vertical splitting, table merging and column mapping | x | x |
| Establishes a logical and physical model upon completing a reverse engineering of a database or script file | x | x |
| Performs standard validation checks for logical and physical models | x | x |
| Validation while on-screen editing indicates to the user that standard object field lengths have been exceeded while inputting entity/attribute names on screen | x | x |
| Object commenting on model objects in modeling environment and web portal to communicate workflow and object statuses | x | x |
| Subject Area Management | | |
| Breaks down large diagrams into smaller subject areas and automatically updates changes to the main model | x | x |
| Nest subject areas within other subject areas | x | x |
| Rearrange the submodel heirarchy via drag and drop capability | x | x |
| Automatically add related objects to a submodel/subject area/package | x | x |
| Customize subject areas and submodels with different layouts, colors and font settings | x | x |
| Naming Standards Support | | |
| Implement customized naming standards for logical names, physical names and translation between logical and physical | x | x |
| Encapsulate a version or set of standards in a template/file so they can be reused across models | x | x |
| Includes naming standard utility to apply naming standards to an entire model | x | x |
| Override global naming standards on the entity,table, attribute or column level | x | x |
| Product must provide a way to override global naming standards on an attribute/column level | x | x |
| Freeze object names that cannot be changed under any circumstances (i.e., the object may be implemented in production while other portions of the model are new) | x | x |
| Native XML Schema Support | | |
| Includes native wizard to build custom XML schemas from a logical or physical submodel/subject area | x | x |
| Translate entities into complex type or elements | x | x |
| Translate domains and attributes into elements, attributes or simple types | x | x |
| Translate reference values / allowed values into enumerations | x | x |
| Incorporate naming standards to translate names in XSD target files | x | x |

| GENERAL LOGICAL MODELING | XE | DATA ARCHITECT |
|---|----|----------------|
| Logical Modeling | | |
| Provides a separate modeling environment for logical modeling than physical models | x | x |
| Provides modeling of database views in the logical model in preparation of DBMS-specific model generation | x | x |
| Supports data model fundamentals in propagating Foreign Keys when relationships are established between entities | x | x |
| Diagrammatically hide foreign keys for conceptual presentations | x | x |
| Generate UML class structures from logical entities | x | x |
| Support Logical Versus Physical Nomenclature for objects | x | x |
| Supports Logical and Physical where used information | | |
| Shows how logical entities, attributes and views are represented in each physical model | x | x |
| Visually see submodel or subject area "Where Used" within Entity or Table Editor | x | x |
| Visually see how a logical entity relates to many physical tables in Physical Model(s) | x | x |
| Customize logical and physical mappings between entities and tables and attributes and columns | x | x |
| Navigate between related logical and physical entities/tables | x | x |
| Data Dictionary System | | |
| Support an ability to access and reuse common elements | x | x |
| Establishes Reusable Domain System Across Data Models | x | x |
| Supports Reusable User Defined Type System Across Data Models | x | x |
| Supports a reusable Rule/Constraint system both logically and physically | x | x |
| Includes an internally managed system for Allowed Valued (Reference or Lookup Data) that can be reused across the model | x | x |
| Provides the user with a simple means to display where dictionary elements have been distributed to for impact analysis | x | x |
| Meta Model Extensibility | | |
| Product must be able to support user-defined meta model extensions simply and efficiently | x | x |
| Classify types of extended meta data by object class | x | x |
| Ability to "push" Attached Extended Meta Data to desired objects | x | x |
| Easily see where extended meta data has been bound to, object by object. | x | x |
| Product's object property editors must provide a UI to access extended meta data | x | x |
| Ability to access external source files and launch them for view/edit purposes from within the modeling product itself. | x | x |
| Data Security Management | | |
| Easily capture security metadata | x | x |
| Provides method for classifying the security impact of data | x | x |
| Allows model objects to be mapped to compliance regulations such as SOX or HIPPA | x | x |
| Assign privacy levels of data within a model, submodel, table or column | x | x |
| GENERAL PHYSICAL MODELING | XE | DATA ARCHITECT |
| Physical Modeling | | |
| Connects to datasources through 3rd party ODBC drivers | x | x |
| Connects to datasources through DBMS client software | x | x |
| General Reverse Engineering Functionality | | |
| Provides a list of owners whose objects can be reverse engineered into a physical model | x | x |
| Filter by object type to reverse engineer into a physical model | x | x |
| Filter a list of tables/views to reverse engineer into a physical model | x | x |
| Infer primary and foreign keys during the reverse engineer process | x | x |
| Build a domain list based on columns in the database to help enforce and promote standardization and reuse | x | x |
| Connect to datasources through 3rd party ODBC drivers for forward engineering via ODBC | x | x |
| Connect to datasources through DBMS client software for forward engineering via native client connections | x | x |
| Provides a list of tables/views to reverse engineer into a physical model | x | x |
| Produce a .SQL script based upon selected objects | x | x |
| Produce separate .SQL files for each model object so that they can be place easily into source code systems | x | x |
| Forward engineer selected objects directly to database | x | x |

| | XE | DATA ARCHITECT |
|---|----|----------------|
| Modify database structures based upon changes to model | x | x |
| Diagram updates when changes occur in the database | x | x |
| Push changes up to the logical model from the physical model/database | x | x |
| Data Movement / ETL Management | | |
| Captures ETL mappings and data movement rules | x | x |
| Capture data movement rules to document the behavior of the data in a table when inserted, updated, archived, purged, etc | x | x |
| Capture source column mappings and transform logic/description | x | x |
| Capture target column mappings and transform logic/description | x | x |
| Capture multiple levels of source/target mapping to represent lineage of the data | x | x |
| Visual data lineage that visually documents source/target mapping and sourcing rules for data movement across systems | x | x |
| Capacity Planning Functionality | | |
| Manage and estimate growth of data for a physical model | x | x |
| Store row count info for each table | x | x |
| Reverse engineer growth metrics from live database | x | x |
| Assign different growth rates for each table based on business rules | x | x |
| Allow for multiple growth rate types like "by row" or "by percent" | x | x |
| Parser-support Between Physical Model Objects | | |
| Supports strong parsing technology to establish ties between precompiled database code (stored procedures) and the tables that may be dependant upon them | x | x |
| Automatically detect table dependency from stored procedure code | x | x |
| Provides UI to easily determine object 'dependants' for impact analysis | x | x |
| Propagates updates automatically to code when referenced objects are changed | x | x |
| Allows user to access object CREATE code from individual object editors before code generation utilities | x | x |
| Color Coded DDL Syntax that displays database reserved words/key words in traditional color-coded syntax within the product | x | x |
| Represent physical objects like procedures, packages, functions, tablespaces and their dependencies on the model | x | x |
| Automatically link Database Views to Tables upon reverse engineering | x | x |
| Database Security Objects and Grants | | |
| Reverse and forward engineer database security objects and permissions | x | x |
| Manage database users and associated GRANT statements | x | x |
| Manage database roles and associated GRANT statements | x | x |
| GENERAL REPORTING | | |
| Output model information to RTF-readable formats (like Microsoft Word) | x | x |
| Produce Reports in HTML format | x | x |
| Reports allows externally 'bound' documentation to be displayed directly within the body of the HTML report through OLE technology | x | x |
| Reports include a navigable, legible, read-only version of the data model | x | x |
| Allows navigation to reported meta data by clicking on model objects in HTML Data Model Image | x | x |
| Reports offer a list of objects contained within the report and hyperlink them to their information | x | x |
| Generate model meta data to XLS format | x | x |
| Produce W-3-C Compliant XML and DTD Meta Data Output | x | x |
| *Export model information to BI, ETL, other modeling tools, and industry-standard metadata interchange formats. Available through MetaWizard | x | x |
| *Import model information from BI, ETL, other modeling tools, and industry-standard metadata interchange formats. Available through MetaWizard | x | x |
| REPOSITORY | | |
| Collaboration | | |
| Allows multiple modelers to access models concurrently | x | |
| Notifies modelers connected to Repository Diagrams who is working on same objects | x | |
| Notifies modelers connected to Repository of the status of the collaboration status of an object | x | |
| Includes intelligent conflict resolution system when two or more modelers are contending to change the same object | x | |

| | XE | DATA ARCHITECT |
|--|----|----------------|
| Implements a separate system for implementing common items (Domains, extended properties etc) across diagrams stored in the Repository | x | |
| Provides an interface to see how common dictionary objects are used across the repository | x | |
| Provides a classification system to group diagrams together in the Repository | x | |
| Version Control | | |
| Captures Periodic Releases of Data Models | x | |
| Ability to revert to capture releases (Roll back) | x | |
| Compare and merge information between diagrams in the Repository | x | |
| Supports commenting on check ins and check outs like source control system | x | |
| Security & Privileges | | |
| Implements a system to create unique Repository Users with individual privilege settings | x | |
| Allows levels of security access to diagrams and objects based upon team hierarchies | x | |
| Product security is able to protect diagrams for unwanted access | x | |
| Allows control over certain object types managed in the Repository lower than "Connect" rights | x | |
| Control access to certain re-useable data elements across diagrams from unwanted access | x | |
| Allows the users to check out individual objects, not just the whole diagram by default | x | |
| Allow a user to check out an object and bar others from doing so while user has item checked out | x | |
| Enterprise Data Dictionary | | |
| Support an ability to access and reuse common elements across models | x | |
| Establishes Reusable Domain System Across Data Models | x | |
| Supports Reusable User Defined Type System Across Data Models | x | |
| Supports a reusable Rule/Constraint system both logically and physically | x | |
| Includes an internally managed system for Allowed Values (Reference or Lookup Data) that can be reused across the model | x | |
| Provides the user with a simple means to display where dictionary elements have been distributed to for impact analysis | x | |
| Enterprise Portal | | |
| Browser-based searching (simple and advanced) of ER/Studio Repository Metadata | x | |
| Customized, self-service reporting on ER/Studio Repository metadata | x | |
| Centralized navigation and exploring of models via a browser | x | |
| Object commenting of portal content | x | |
| Labeling of content in repository to classify and group related content | x | |
| Activity monitoring to see what objects have changed in the Repository and users who have accessed the portal | x | |
| GENERAL PRODUCT USABILITY | | |
| N-level Undo / Redo | x | x |
| Provides thumbnail view to navigate large diagrams | x | x |
| Marquee Lasso Zoom | x | x |
| Explorer Browser Object Navigation | x | x |
| Allows user to quickly see the number of entities, attributes, relationships, views etc that are in the model | x | x |
| Property editors conform to Windows standards and allow 1 layer deep access to properties | x | x |
| Property editors conform to Windows standards and allow expansion for ease in entering data | x | x |
| On Screen Object Editing (Editorless via Key Strokes) | x | x |
| On-Screen Logical Primary Key Creation (Editorless via Key Strokes) | x | x |
| On-Screen Attribute Copy/Move Function | x | x |
| Global Search/Report/Replace Utility | x | x |
| Wizard-driven Task Completion | x | x |
| Lasso Multiple Objects and access Right Mouse Options | x | x |
| Offers simple and fast way to break down large models by lassoing desired objects and quickly establishing a subject area of them | x | x |
| Quick access to diagrammatic property changes to desired objects like color | x | x |
| Navigate user to desired Help section from specific property editors, etc. | x | x |

| | XE | DATA ARCHITECT |
|--|----|----------------|
| Non-Proprietary Automation Interface (API) | | |
| Provides a programmatic interface in a common & industry-accepted language in order to programmatically access product's object model | x | x |
| Support sVB or VBA-like macro creation | x | x |
| Near-immediate accessibility to macros to ensure workflow and productivity | x | x |
| Macro editor within product provides 'keystroke access' to product's object model for quick reference and accuracy | x | x |
| Provide a reference map of the products objects | x | x |
| Sample scripts to use as a basis for user macros included | x | x |
| Variety of different layout strategies for logical and physical models | x | x |
| SOFTWARE MODELING | | |
| Standard with sample projects to familiarize users with features | x | |
| Sample cheat sheets with interactive tutorials | x | |
| Query/View/Transformation language to transform UML, BPMN, data models and custom model types. | x | |
| Logical and physical packages to group elements and store diagrams | x | |
| Model shortcuts for creating multiple shortcuts to the same element on different model diagrams. | x | |
| Model Hyperlinking to create hyperlink from diagrams to other system artifacts and browse them directly. | x | |
| Interoperability is supported with various types of model import and export to XMI, MDL and MDX. | x | |
| External documentation for open projects. Output formats for RTF, HTML, TXT, PDF and XSL-FO | x | |
| Supports UML 2.0 to visualize, specify, construct, and document the artifacts of the distributed objects systems. | x | |
| Optional profile to support the "modeling in color" methodology with support for roles, moment-interval, Mi-detail, party, place, thing and description. | x | |
| Supports the most frequently used diagrams and notations defined in the UML 2.0 specification, including activity, class, use, component, composite, deployment, state machine and interaction diagrams. | x | |
| Includes pre-installed profiles and allows users to create profile definitions, including profile definition projects such as stereotypes, palette contributions, extensions and contributions. | x | |
| Supports two-way and three-way EMF and UML model comparisons in a tree view. Results can be exported to an EMF XMI file. | x | |
| Utilizes standard Eclipse synchronization APIs to provide integration with version control systems to compare and merge shared models. | x | |
| Supports templates to provide the ability to show templates, template signatures, parameters and template bindings in a UML 2.0 diagram | x | |
| Object Constraint Language (OCL) 2.0 support for syntax highlighting, error validation, code completion and model queries. | x | |
| Design patterns that are available in stock patterns supporting Gang of Four, J2EE Design, Fowler's EAI, and Web Services, and custom design patterns. | x | |
| OCL-based model audits and metrics support model inspections and can be easily be defined, saved, and reused. | x | |
| Version control systems in place to enable multiple users to work with one modeling project. Supports version control systems that can be integrated into Eclipse. | x | |
| BUSINESS PROCESS AND CONCEPTUAL MODELING | | |
| Support for high-level conceptual modeling using elements such as subject areas, business entities, interactions, and relationships | x | |
| Model links between any conceptual or process modeling elements allowing you to trace relationships between models | x | |
| Conceptual models can be exported to Embarcadero ER/Studio to become the foundation for creating ER/Studio logical data models | x | |
| Support for straightforward process modeling that uses standard elements such as sequences, tasks, swim lanes, start events, and gateways | x | |
| Optional automatic validation of process diagrams to ensure compliance with the BPMN specification and prevent the addition of non-compliant modeling elements | x | |
| Independent sub-processes and embedded collapsible sub-processes can be included within a business process to allow for maximum flexibility in diagramming, while still ensuring a workable visual diagram | x | |
| Impact analysis reports can be generated to show interrelationships between process, data, stewardship, business rules, diagram usage, heritage, connecting objects etc. | x | |
| Impact analysis reports can filter based on type of relationship, object type, or text strings including wildcard matching | x | |

Download a Free Trial at www.embarcadero.com

Corporate Headquarters | Embarcadero Technologies | 100 California Street, 12th Floor | San Francisco, CA 94111 | www.embarcadero.com | sales@embarcadero.com