

# ODE DATABASE STANDARDS



## DOCUMENT REVISION LIST

Version #	Revision Date	Revision Description	Author
2.5	11/17/2008	Minor revisions and updates	M. Carmack
2.4	5/18/2007	Removed standards acceptance form, which is now a separate PMO artifact	M. Meredith
2.3	3/1/2007	Minor revisions to software versions	M. Carmack
2.2	4/4/2005	Several clarifications added to chapters 5, 7	M. Carmack
2.1	12/11/2003	Update to Appendix A.	M. Carmack
2.0	11/7/2003	Major revisions and additions	M. Carmack
1.7	2/26/03	Made the requested updates to 8.4.2	Ralph Mohr
1.6	2/10/03	Revised Format Topic	Ralph Mohr
1.5	2/03/03	Updated Transform Topic	Ralph Mohr
1.4	1/15/03	Revise based on feedback from review session	Ralph Mohr
1.3	1/07/2003	Added physical standards, sections 8 and 9	Ralph Mohr
1.2	12/10/2002	Revised based on Don Kingery review feedback	Ralph Mohr
1.1	12/01/2002	Revised based on review feedback	Ralph Mohr
1.0	11/12/2002	Initial Release	Ralph Mohr

## Table of Contents

<b>1.</b>	<b>INTRODUCTION.....</b>	<b>1</b>
<b>2.</b>	<b>ROLES AND RESPONSIBILITIES.....</b>	<b>2</b>
2.1.	Business Subject Matter Experts (BSME) .....	2
2.2.	Technical Subject Matter Experts (TSME) .....	4
2.3.	Logical Data Modeler.....	5
2.4.	Physical Data Modeler.....	6
2.5.	Database Administrator (DBA) .....	6
<b>3.</b>	<b>DATABASE ENVIRONMENT .....</b>	<b>8</b>
3.1.	Database Management System (DBMS) .....	8
3.2.	Client Connectivity Software.....	9
3.3.	Query Software.....	9
<b>4.</b>	<b>DATA MODELING ENVIRONMENT .....</b>	<b>10</b>
4.1.	Data Modeling Software .....	10
4.2.	Data Model Repository .....	10
4.3.	Data Model Conventions .....	13
4.4.	Data Model Development Process .....	13
4.5.	Data Model Management Change Control Process .....	14
<b>5.</b>	<b>GLOBAL METADATA STANDARDS .....</b>	<b>16</b>
5.1.	Data Definition Standards.....	16
5.2.	Subject Area Standards.....	17
5.3.	Text Box Standards .....	18
5.4.	Presentation Standards .....	18
5.5.	Relationship Standards .....	18
5.6.	Invoking Standards within ERwin .....	19
<b>6.</b>	<b>LOGICAL DATA MODEL STANDARDS.....</b>	<b>21</b>
6.1.	General Logical Naming Standards.....	21
6.2.	Logical Data Model Entity Standards .....	22
6.3.	Logical Data Model Attribute Standards .....	24
6.4.	LDM Relationship Standards.....	27
<b>7.</b>	<b>PHYSICAL DATA MODEL STANDARDS.....</b>	<b>30</b>
7.1.	Physical Naming and Abbreviation Standards .....	30
7.2.	Physical Data Model Table Standards .....	30
7.3.	Physical Data Model Column Standards.....	32
7.4.	Foreign Key Standards.....	33
7.5.	Index Standards .....	33
7.6.	Other Physical Model Object Standards.....	35
7.7.	Documenting Logical to Physical Model Transitions .....	38
<b>8.</b>	<b>OTHER PHYSICAL DATABASE STANDARDS.....</b>	<b>40</b>
8.1.	DBMS Environment.....	40

8.2. Unmodeled Physical Objects.....	41
<b>APPENDIX A: DOMAIN DICTIONARY .....</b>	<b>45</b>
<b>APPENDIX B: DATABASE PROJECT PLAN TEMPLATE.....</b>	<b>48</b>
<b>APPENDIX C: LOGICAL DATA MODEL REVIEW FORM.....</b>	<b>53</b>
<b>APPENDIX D: PHYSICAL DATA MODEL REVIEW FORM.....</b>	<b>57</b>

## 1. Introduction

This document defines the database standards in place at the Ohio Department of Education. All new development efforts that include an enterprise database component must adhere to these standards.

Each standard usually falls into one of two broad categories:

### **Data Modeling Standards**

Formal data modeling is performed to:

- Capture the collective knowledge about Ohio Department of Education's enterprise data in a common repository
- Reduce application development time by providing a "map" against which data requirements can be assessed
- Establish control over data practices so that data integrity is established and maintained
- Establish consistency over data definitions and data usage by documenting a commonly accepted set of business definitions and business rules
- Reduce data redundancies and eliminate data inconsistencies across systems
- Allow sharing of information across the organization
- Promote the reusability of data assets and processes
- Provide a well-documented data design that can serve as a solid foundation for physical database design optimization tasks
- Document the design of the physically implemented database for an application in a graphical and textual manner, providing improved documentation for application developers and database administrators.

ODE's data model management strategy recognizes that all data models developed to support operational and decision support projects are valuable intellectual assets of the Ohio Department of Education. Once developed, the data models become part of the intellectual capital of the organization and should be safeguarded as other department assets are.

### **Database Lifecycle / Change Control Standards**

ODE has a strongly defined database lifecycle plan to control development, testing, and production implementation of new databases and changes to existing databases.

When anything in the document is not explicitly stated or is unclear, always err on the side of the cautious and ask an ODE DBA for clarification. The author is not a lawyer and this document is not a contract. Debate over what the standards "say" versus what they "meant" will not be entertained; the intent of the standards override any assumptions a developer may try to make. Any questions a developer offers are very welcome and will help to make this document clearer over time.

## 2. Roles and Responsibilities

This document provides standards for data modelers, DBAs, developers, and project managers working for the Ohio Department of Education as an employee or as a contractor.

- Data modelers and database administrators (DBAs) need to understand the data model management process and the specific standards for their area of responsibility.
- Application developers need to understand how to use a data model and understand the impact of data model changes on their programs.
- Project managers need to understand the data model and database life cycle so it can be incorporated in the overall project life cycle.

Although data modelers and DBAs create all data models, close interaction with Business Subject Matter Experts (BSMEs) and Technical Subject Matter Experts (TSMEs) is required to produce high quality and useful databases.

This document assumes a basic understanding of logical and physical data modeling concepts and terminology. The work product created by the Logical and Physical Data Modelers will be used by a broad range of technical and business users, making their effort a critical deliverable within the Ohio Department of Education.

Specific roles required to support the implementation of the database management strategy are defined below. A role represents a function executed by a person. Not all roles within the organization will have a one-to-one relationship with a person. An individual may be called on to fill multiple roles during the development of a data model or to fill several roles simultaneously. Some resources may become permanent members of a data management staff, while others serve as temporary resources supporting a specific iteration of the data model's development. The number of individuals required for a data management team varies according to the size of the size and complexity of the database being created.

The ODE database management strategy identifies the following roles and responsibilities:

### 2.1. Business Subject Matter Experts (BSME)

Logical data model development and physical data model development requires participation of Business Subject Matter Experts. Each BSME must be knowledgeable about a specific subject area or areas within their business area. An example of a BSME is an individual within Fiscal that has in-depth knowledge of the state budgeting process.

#### **BSME Qualities**

BSMEs are the primary conduit of information transfer from the business office to a data modeler. To ensure that this process results in an accurate and effective data model, a BSME should possess certain characteristics:

- A thorough knowledge of business rules for their area
- A strong understanding of the data used in their area
- Strong interpersonal and communication skills

- A basic understanding of IT concepts, especially the information required for metadata
- Comfortable with and respectful of iterative processes
- Empowered to make data model decisions for their business area. The ability to create an enterprise view requires that the BSMEs negotiate and reach agreements for data definitions and business rules between lines of business.

In most cases an office's Data Manager will be a BSME, and quite likely the primary BSME. Other personnel can be included as the office and/or data modeler deems appropriate.

### BSME Roles and Responsibilities for Logical Data Model

Roles and Responsibilities	Examples
Developing Standard Terminology	Finalizing the use of the term <b>VENDOR</b> instead of the more ambiguous term <b>SUPPLIER</b> .
Developing Data Definitions	<b>VENDOR</b> : An external party that acts as a source for merchandise offered for sale or for supplies consumed by the Ohio Department of Education.
Developing Business Rule Definitions	A <b>PURCHASE ORDER</b> is associated with only one <b>VENDOR</b> . A <b>VENDOR</b> is associated with zero, one or more <b>PURCHASE ORDERS</b> .
Developing Business Name Aliases	A <b>VENDOR</b> may also be called a <b>SUPPLIER</b> when referring to an entity that provides goods and/or services to Ohio Department of Education
Developing Business Constraints	One operating <b>DISTRICT</b> may only be contained in one operating <b>REGION</b> .
Defining Valid Values	<b>LOCATION TYPE</b> can only be Work or Residential
Assumes Ownership of Specific Attributes	Accounts Payable Managers are responsible for <b>VENDOR</b> data attributes such as Freight Policy and Payment Terms.  Purchasing Managers are responsible for <b>SHIPMENT</b> data attributes such as Shipment Date and Transportation Method.
Communicates Data Model Issues and Status to their Departments	Because practices vary by business, the data model may show that <b>VENDOR INVOICES</b> are paid by line item whereas in a particular business <b>VENDOR INVOICES</b> are paid in their entirety.

### BSME Roles and Responsibilities for Physical Data Model

Roles and Responsibilities	Examples
Developing Data Properties	VENDOR NAME will not exceed 60 characters.
Assumes Joint Ownership of Specific Data within Table Columns	Accounts receivable would ensure the validity of ORGANIZATION numbers.
Developing Aggregation Processes	STORE sales data is aggregated to Store Level by Week, Month, Quarter and Year.
Developing Data Retention Criteria	ENROLLMENT FORMs must be kept for a period of a rolling 36 months after the ACADEMIC YEAR has been closed.
Defining Standard Calculations	Total School Staffing = Administrative Staff FTE + Classroom Teaching Staff FTE
Defining Source Data	Attendance Rate calculations use the EMIS year end data elements "Attendance Days", "Absence Days", and "Unexcused Absence Days".
Defining Data Security Specifications	A Superintendent can make updates to their District's OEDS-R data. The Superintendent can see data for other districts but does not have the ability to enter or update information for those districts.
Estimating Volume Metrics	Identify how many Students are enrolled in a school within a given year.

### 2.2. Technical Subject Matter Experts (TSME)

Physical data model development requires the participation of Technical Subject Matter Experts. At the Ohio Department of Education, the role of the TSME is filled by data managers, application developers, and database administrators. Each TSME must be knowledgeable about a specific subject area or areas within their business and application area. An example of a TSME is an individual within the Ohio Department of Education that has in-depth knowledge of the systems used to support the business function.

#### TSME Qualities

- Demonstrated technical skills with the software tools and practices being used
- A thorough understanding of how the agency's, center's, and/or office's applications work
- A strong understanding of the data used and processed in the applications
- An understanding of their source system, especially what information is required for metadata
- Comfortable with and respectful of iterative processes
- Strong interpersonal and communication skills

## TSME Roles and Responsibilities for Physical Data Model

Roles and Responsibilities
Validating Data Properties
Identifying Data Dependencies
Researching business rules and policies that may be reflected in existing systems but have not been documented
Data analysis to confirm each data field is different, that there are no data field duplicates
Validating business rules, as implemented in their application systems
Validating standard calculations, as implemented in their application systems
Validating valid values through existing reference tables and application logic
Validating business constraints, as implemented in their application systems
Validating aggregation processes, as implemented in their application systems
Validating data latency criteria, as implemented in their application systems
Validating data sources, as implemented in their application systems

### 2.3. Logical Data Modeler

The Logical Data Modeler role is often one of the most misunderstood in the database lifecycle. A Logical Data Modeler has the critical responsibility of creating a “picture” of what the data looks like from the business perspective, independent of any specific applications or technologies that implement the data system.

A Logical Data Modeler should possess the following characteristics:

- Demonstrated skill in data modeling and data architecture.
- Functional knowledge of the business data being modeled, or the ability to obtain that knowledge through communications with BSMEs and existing technical documentation.
- An understanding of metadata and its importance to enterprise data modeling strategies
- Experience with and the skill to effectively use ODE’s standard data modeling software (ERwin)



**Note:** Contractor teams often assign this role to their single DBA resource because it is “about the database”. Be aware that “pure” DBAs can sometimes bring too much focus on the physical database into this process, sometimes even questioning the need for a logical data model in the first place! This disconnect can cause unexpected delays, so please assign this role appropriately.

In the transformation from the logical data model to the physical data model, the Logical Data Modeler provides support to the Physical Data Modeler role in the form of knowledge transfer regarding why certain decisions were made.

It is common for one person to assume both the role of Logical Data Modeler and Physical Data Modeler. This is fine as long as that person understands the differences between the two types of data models.

## 2.4. Physical Data Modeler

The individual filling the role of a Physical Data Modeler has skills in data modeling as well as a strong technical background in designing physical databases, particularly for the database management system (DBMS) selected for the specific application. The Physical Data Modeler is responsible for converting the logical data model into the optimized (denormalized, indexed etc.) physical data model with guidance from the application developers and knowledge of system performance requirements. He/she also sees that appropriate tables within a physical data model are fully documented and mapped to the appropriate source system. When the Physical Data Modeler completes this task, all tables, columns, indexes and other DBMS element properties are fully defined and documented. The Physical Data Modeler also assists the Database Administrator in capacity planning.

A Physical Data Modeler should possess the following characteristics:

- Demonstrated skill in data modeling and data architecture.
- Experience designing objects for and using the target database platform
- An understanding of metadata and its importance to enterprise data modeling strategies
- Experience with and the skill to effectively use ODE's standard data modeling software (ERwin)

It is common for one person to assume both the role of Logical Data Modeler and Physical Data Modeler. This is fine as long as that person understands the differences between the two types of data models.

## 2.5. Database Administrator (DBA)

The individual filling the DBA role has a strong technical background in managing databases under the DBMS selected for the specific application. The DBA has a strong understanding of physical data models, and some familiarity with logical data models. The DBA may assist in developing the physical data model. The DBA is responsible for capacity planning, security considerations, and developing and implementing the database design. The DBA also manages the implementation of the database once in production, and administers data retention and recovery processes.

While an application is being created, the DBA role is subdivided into two categories:

- **Contracted DBAs** are the database resources that a contracted team of consultants supplies to work on the project. They provide application development support in the development environment.
- **ODE DBAs** are employees of ODE. ODE DBAs are ultimately responsible for all databases, and only ODE DBAs can perform system level tasks or modify database

structure in the QA and production environments. An ODE DBA must approve all data models created by Contracted DBAs.

### **3. Database Environment**

#### **3.1. Database Management System (DBMS)**

##### **3.1.1. Oracle**

ODE's standard database platform is Oracle. All new databases for web applications will be implemented in Oracle.

No database or application design may depend on features introduced in a version of Oracle higher than the current ODE standard. In other words, no design is permitted to require a database upgrade. The current version of Oracle in standard use at ODE is Oracle 10g Release 2.

ODE's Oracle environment runs on HP-UX Unix servers with PA-RISC processors. Only ODE DBAs have direct access to the Unix servers. Contracted DBAs and application developers will access Oracle over the network via a SQL\*Net connection.

##### **3.1.2. SQL Server**

SQL Server is supported by ODE in legacy mode for web applications. Only enhancements or expansions of existing SQL Server databases are permitted. No new SQL Server databases are to be designed. When an application powered by a SQL Server database is reengineered (e.g. to change from ASP to .NET), the database should be migrated to Oracle.

SQL Server is also supported when third-party software requires a SQL Server repository. This is not an uncommon scenario, e.g. all Microsoft server products that require a database repository (e.g. Sharepoint) are compatible only with SQL Server. New SQL Server databases are permitted in this scenario by necessity. However, whenever Oracle is supported as a repository DBMS, Oracle must be used. Additionally, when evaluating competing software products and comparing the "pros and cons" of each, requiring SQL Server should be treated as a "con" where support for Oracle should be treated as a "pro".

##### **3.1.3. Other DBMS Products**

No DBMS product not previously mentioned will be considered for enterprise deployment and support.

##### **3.1.4. Desktop Database Products**

ODE supports desktop database products (such as Microsoft Access or FileMaker Pro) for personal productivity only. No enterprise application should include or depend on the presence of a desktop database product on any user's PC. Further, Database Services does not officially support any personal databases designed and implemented in a desktop database product.

### **3.2. Client Connectivity Software**

The Oracle Client software will be provided to contractors and staff to enable SQL\*Net connections to Oracle databases.

The Oracle ODBC driver will also be provided. Only the Oracle ODBC driver should be installed because it provides the most access to Oracle's full feature set. Do not use another vendor's driver (e.g. Microsoft's ODBC driver for Oracle) without consulting with Database Services. Such drivers will be approved when there is a strong technical case, e.g. when a software package is only supported when using a specific ODBC driver.

Oracle database names are resolved using Oracle Internet Directory (OID). No TNSNAMES.ORA files are used.

For developers writing .NET applications, the ODP.NET software will be provided.

### **3.3. Query Software**

The TOAD query tool will be provided to contracted DBAs. Subject to license availability it may also be provided to contracted developers if they can demonstrate a need. However, it is expected that developer needs will be wholly satisfied within Visual Studio and/or by Oracle SQL Developer and iSQL\*Plus.

Oracle SQL Developer is a Java-based graphical query tool similar to TOAD. It is available as a free download on Oracle TechNet (<http://www.oracle.com/technology>).

iSQL\*Plus is a web-based query tool available to anyone on ODE's network. No client software other than a web browser is required.

## 4. Data Modeling Environment

### 4.1. Data Modeling Software

ODE has adopted ERwin 4.1 as the standard tool for logical and physical data modeling. All data models must be developed in ERwin.

One copy of ERwin will be provided to each contracted data modeler. To preserve licenses for data modelers, ERwin is not provided to application developers and project managers who would use the tool to simply browse data models. The data modeler can generate HTML versions of data models in development to share with team members that do not have a copy of ERwin.

### 4.2. Data Model Repository

ERwin data models must be saved in the ERwin ModelMart repository. This requirement serves several purposes:

- Data models are protected in a database included in enterprise backup processes
- Data models are automatically versioned
- Data models, or portions of a data model, can be locked when checked out to prevent multiple data modelers from making conflicting changes
- There is one agreed upon “master” data model

All data modelers provided with a copy of the ERwin data modeling software will also be provided with an account in the ModelMart repository. If the data modeler does not know how to use ModelMart, he should request an overview from an ODE DBA.

#### 4.2.1. Libraries

ModelMart supports a library structure for the creation, storage, and maintenance of ODE data models. A basic library structure is used to support data models for ODE.

**Development:** Development libraries are used to store data models that are either being created, being enhanced in a new development iteration, or being modified as a solution to production problems.

Each data model will have its own development library to support multiple models being developed in parallel, i.e. a *primary data model* and one or more *alternate data models* for testing different methods or implementations. Data modelers are free to create as many alternate data models as suits their need, but only the primary data model will be promoted into the QA library.

For new database projects, an ODE DBA will create the development library and grant access to the data modeler. For enhancements to existing data models, an ODE DBA will grant access to the existing development library to the data modeler.

**QA (Test):** The QA library is used to store completed data models that are being tested and evaluated for movement into production. The QA library is used to

support the change management procedure. All data models share a single QA library.

The primary data model in the development library must be reviewed and approved by an ODE DBA before being promoted into the QA library. Only an ODE DBA can store a data model in the QA library. All ModelMart users can read data models in the QA library.

**Production:** The production library houses the completed, tested, reviewed, and approved data models that support the databases in the production environment. The data models stored and maintained in the production library must accurately map to the tables that exist in the production database. All data models share a single production library.

The data model in the QA library is the only data model that can be promoted to the production library. It is not permitted for any data model to go directly from development to production. Only an ODE DBA can store a data model in the production library. All ModelMart users can read data models in the production library.

## 4.2.2. Data Model Names

### 4.2.2.1. Primary Data Models

A data model's primary name must be simple and immediately identify the topic of the data model. Since most data models represent application systems and/or processes, data models names are usually derived from the application system name or acronym.

Examples of good primary data model names are:

- OEDS-R
- EMIS\_ODS
- CCIP

### 4.2.2.2. Alternate Data Models

Alternate data models can be used by a data modeler to test different methods or implementations, or to “work ahead” in an iterative development cycle when the primary data model cannot be modified because it is under formal review for promotion to QA. They are temporary by design because any work to be preserved must be merged into the primary data model eventually.

An alternate data model's name should consist of the primary data model name, plus one or more keywords to identify the alternate use of this particular model to the data modeler. Some examples of alternate data model names are:

- OEDS-R\_legacy
- EMIS\_ODS\_TestNewFeature
- CCIP\_phase4

#### 4.2.2.3. Versioning

When data models are saved back to ModelMart correctly by clicking the  button, ModelMart automatically creates a new version of the data model name. Information about that version of the data model (version number, who created the version, and when the version was created) is attached to the data model and is viewable in the ModelMart Version Manager (on the *ModelMart* menu in ERwin).

Examples of ModelMart version names for a data model named “OEDS-R”:

OEDS-R: v.3 by user modeler1 on Feb. 12, 2003  
OEDS-R: v.2 by user modeler2 on Jan. 20, 2003  
OEDS-R: v.1 by user modeler1 on Jan. 06, 2003

A data modeler should never attempt to manually duplicate or replace version functionality with alternate data models. I.e., do not perform a “Save As” operation, appending version numbers or dates to the original model name. This disrupts ERwin’s built-in version functions.

Developers should create a new version of a work-in-progress database (i.e. save it back to ModelMart) at least once a day.

#### 4.2.2.4. External Data Models (.er1 Files)

As a rule, all ODE data models must be stored and maintained in the data model repository. When special circumstances require that a data model be developed outside of the data model repository (e.g. short term off-site development without VPN access to ODE’s network), the data model must be saved to disk as an ERwin “.er1” file. A naming standard has been created to support these unique cases.

External data model names are in the format of *Name vX\_Y\_ yyyymmdd.ER1*, where:

- *Name* is the primary data model name in ModelMart
- A lower case *v* signifies “version”
- *X* represents the major version number, assigned based on significance “release” levels
- *Y* represents the minor version number
- *yyymmdd* identifies the date the data model was saved/updated
- *.ER1* is the standard extension for all ERwin data models

Note that the version numbers in this naming convention are not related to the ModelMart version numbers in any way. They exist to allow a data modeler to simulate versioning with external data models by manually numbering the versions. This is a recommended best practice for external data modelers, but will not be enforced. Eventually the external data model must be checked back into ModelMart, and that is the only version of the model that ODE will use.

### 4.3. Data Model Conventions

The Ohio Department of Education has adopted the Information Engineering (IE) convention as opposed to the Integration Definition for Information Modeling (IDEF1X) for use with both logical and physical data models.

### 4.4. Data Model Development Process

The successful development and maintenance of the data architecture requires participation from both the business and technical communities. Identifying requisite roles and responsibilities and securing qualified resources make a significant contribution to a successful data architecture implementation. Development of and changes to LDMs should be driven by clearly documented business requirements. The data models should reflect a consensus view of the business community and should not be biased to a single business group within the organization. After the business requirements are reflected in the LDM, the PDM should be developed from the LDM. Subsequent to that, the physical database should be generated from the PDM.

This closed-loop model management approach ensures the actual database(s) reflect all business requirements documented in the data models. Any database modifications must flow through the LDM and PDM before they are applied to the database. This includes additions and changes to columns designed to capture operational data that may only be reflected in the physical data model. Although operational data and other “physical only” attributes may only exist in a PDM, a review of the LDM is still required in order to verify the absence of any impact to the LDM. The discipline of always evaluating the logical data model prior to evaluating the physical data model ensures the logical and physical data models retain integrity. The closed-loop data model management approach also ensures the additional benefit of keeping the PDM and databases synchronized, because the database definition language (DDL) to create and manage DBMS objects is always generated from the PDM.

The following factors may result in the creation of new subject areas or modifications to existing subject areas:

- Changes to LDM to support:
  - Changes in business rules due to policy or legislative mandates
  - Changes in the scope or functionality of transaction processing or decision support system projects
  - Discovery of previously undocumented business processes
- Changes to the PDM to support:
  - Changes in the logical data model
  - Database performance tuning requirements
  - Changes to operational statistics used for system audits such as date and time stamps

#### 4.5. Data Model Management Change Control Process

To ensure data model integrity, standard change control processes will be used. Managing the changes to a data model requires consideration of multiple factors, including version control, configuration level (i.e., development, test, and production), data model sharing among multiple data modeling teams, and applications development.

Data model integrity and adherence to standards will be enforced through a data model review process. The Data Architect will be the owner of this process. The Model Repository Administrator will be responsible for the promoting to a production status only those models that meet ODE data model standards (as spelled out in the model management strategy) and that have been certified through the data model review process.

All data models will be stored in a secured environment. The selection of Data Models from the secured environment and the promotion of data models into the secured environment require specific reviews and approvals. General steps in the data model development process are identified below. Specific change control points are identified in bold.

Responsible Party	Participants	Responsibility
Database Services Manager		Assign Logical Data Modeler to development project
Database Services Manager		Assign Physical Data Modeler to development project
AllFusion ModelMart Administrator		<ul style="list-style-type: none"> <li>• Create required LDM subject area (if required)</li> <li>• Assign specific subject areas to the Logical Data Modeler</li> </ul>
Logical Data Modeler		Assume responsibility for the assigned Subjects Areas
Logical Data Modeler	BSMEs, TSMEs, DBAs	<ul style="list-style-type: none"> <li>• Evaluate data requirements</li> <li>• Develop proposed LDM</li> </ul>
Logical Data Modeler	BSMEs, TSMEs, PDMR, Project Manager, Data Management Group Manager	<ul style="list-style-type: none"> <li>• Conduct model review session to review proposed LDM</li> </ul>
Data Architect	BSMEs, Project Manager, Data Management Group Manager	Approve proposed LDM
Logical Data Modeler		Implement approved LDM subject area into production
Physical Data Modeler		Assume responsibility for the assigned approved PDM subject area
Physical Data Modeler	LDMR, TSMEs	<ul style="list-style-type: none"> <li>• Evaluate data performance requirements</li> <li>• Develop proposed PDM</li> </ul>
Physical Data Modeler	TSMEs, LDMR, Project Manager, DBA, Data	Conduct model review session to review proposed PDM

Responsible Party	Participants	Responsibility
	Management Group Manger	
Data Architect	TSMEs, Project Manager, Database Management Department Head	Approve proposed PDM
Physical Data Modeler	DBA	Turn-over PDM to DBA for implementation and on-going management

Logical Data Modelers, Physical Data Modelers and Database Administrators are responsible for managing data model changes. They are responsible for working with other Logical and Physical Data Modelers, Database Administrators, and project team members to ensure that conflicts or differences between data model versions that arise during the merge and update processes are correctly resolved. To assist them, data model review sessions will be implemented as the cornerstone of the Data Model Management Change Control Process. A data model review session is similar in nature and function to a design review or code walk-through. Data model reviews should occur when a new subject area is added or a current subject area is enhanced.

The data model review sessions should include the following participants. In the review session, the participants have the following responsibilities:

<u>BSMEs</u>	Ensure the LDM accurately represents business rules, processes, and definitions
<u>LDMRs</u>	Present the LDM to the BSMEs for review
<u>PDMRs</u>	<ul style="list-style-type: none"> <li>Evaluate ramifications to current physical database structures and estimate impact of proposed LDM changes</li> <li>Present the PDM to the TSMEs for review</li> </ul>
<u>DBAs</u>	Evaluate ramifications to current systems and estimate impact of proposed PDM changes
<u>TSMEs</u>	Confirm Logical and Physical Data Models will satisfy user requirements
Project Manager, Data Architect, Data Management Group Manager	<ul style="list-style-type: none"> <li>Ensure standard practices are adhered to</li> <li>Resolve problems, as needed</li> </ul>

## 5. Global Metadata Standards

Standardized and integrated data models simplify the development of common interfaces between systems and enable the organization to develop and leverage common data and procedures. Metadata is generally defined as “data about data”. Metadata is an important part of the data model because it documents definitions and business rules according to the organization’s data requirements. Metadata documents the following information about data:

- Describes location of data
- Defines sources of data
- Provides business English descriptions for data being used
- Identifies data types
- Identifies business rules related to the data
- Identifies data sources
- Identifies data stewards

Metadata is an important tool for business users in developing, executing, and understanding queries against the business system. It is also an important tool for the technical staff in developing and maintaining business systems. Metadata allows the business user and technical community to understand business system impacts when enhancements are required. It is generated from and utilized throughout the systems development life cycle.

Complete and accurate metadata will be a prerequisite before logical and physical data models are promoted to production status and prior to the generation of production databases. Metadata completeness and quality will be major considerations during the data model review process. Descriptions of all metadata are included in the Logical and Physical Data Model Standards sections of this document.

All metadata related to data such as definitions for entities, attributes, tables, etc., would come from a single entry point, the data model.

### 5.1. Data Definition Standards

A definition must convey complete and accurate information about the object<sup>1</sup> being defined. The definition that accompanies a data model object (i.e., Entity, Attribute, Table, Column, Index, Relationship, etc.) is a piece of metadata that provides important information to all users (business or technical) of the object. Data definitions should adhere to the following rules:

- **The terminology used in a definition must not be a restatement of the object name.** A definition must convey the meaning of the object within the context of the appropriate application. For example, a definition for the attribute ‘Time Zone Code’ of “code that defines the time zone” is not acceptable because it provides no explanation about what a time zone is or why it is important to document this fact.

---

<sup>1</sup> The term “Object” refers to an item in a logical or multi-dimensional data model or physical database, such as: Entity, Attribute, Relationship, Table, Column, Index, Primary Key, Dimension, Member, etc.

- **The definition should use words that are self-explanatory and complete.** It should not be necessary to refer to other definition(s) to obtain the meaning. It should be as “self-documenting” as possible.
- **A definition should not assume specific technical or business experience by the reader.** Someone unfamiliar with the application should be able to reasonably understand the definitions and use them as a source of documentation. Avoid the use of jargon.
- **A definition for an object must be the same in all places where used.** An attribute that appears in multiple entities in a data model cannot have multiple meanings. In fact, it is desirable that an object that appears in multiple distinct and generally unrelated data models maintain a consistent definition across all data models. For example, a “District” means exactly the same thing to ODE regardless of the data model in which it appears.
- **Examples or allowed values should not be included in the definition.** There are standard UDPs defined for metadata of this type.
- **When referring to an entity, the entity name should be in all upper case letters.** This immediately tells the reader that the concept being discussed exists elsewhere in the data model.

## 5.2. Subject Area Standards

A Subject Area in this document refers to a large, logical group of entities, organized to represent all or a portion of a specific business functional area. ERwin supports the creating of subject areas and their use is mandatory.

- **Subdivide a data model into subject areas to create an easier to comprehend data model.** Each subject area should be based on a single business concept, such as Employees or Accounts Receivable. It is a good idea to keep the number of entities within a subject area small enough so that a user can easily view the subject area data model over the web.
- **Subject area names should follow the general logical data model naming standards.** Reference the LDM Data Name Standards (section 6.1) and apply them to data model names and subject area names.
- **The name of the subject area should suggest the types of entities contained in the subject area.** Often the subject area name is the name of a business function or an area of special interest to a group of business users.
- **A subject area name should be in uppercase when the subject area is considered a universal or commonly used data grouping. Use mixed case for application specific subject areas.** For example, ORGANZATION and PERSON are subject areas owned by the OEDS system, but they are enterprise concepts commonly used in other data models and should therefore be contained in an uppercase-named subject area.

- **Special subject areas:**

- REFERENCE ENTITIES contains all entities used as code tables. Individual reference entities may also appear in other subject areas where they provide additional clarification about the subject area.
- UNREVIEWED contains entities that do not yet belong in a common subject area or a business subject area. When the data model is ready for production migration this subject area should be empty.

### 5.3. Text Box Standards

Text boxes in data models should be avoided except in very specific instances. Text boxes are typically forgotten, rendering them obsolete and confusing. Text boxes may be used for:

- **Legends.** When colors are used in data models, a text box may be used to indicate what the colors represent.
- **Interim reminders about status.** These text boxes must be removed when the data model is submitted for review.

### 5.4. Presentation Standards

The use of visual cues, such as special fonts and colors, add clarity and organization to data models. Special visual cues will be used only when their use significantly enhances the visual data model and only when differentiation is desirable but not required.

#### Color

Color will be used to differentiate major entity groups.

Entity	Yellow
Reference Entity	Grey
Future Implementation	Pink
Correlation	Aqua
Supertype-Subtype Relationship	Yellow

#### Fonts

Entity Name	Arial, bold font, 12 point, black, all capitalized
Attribute Name	Arial, regular font, 8 point, black, mixed case
Relationship Name	Arial, regular font, 8 point, blue, lower case
PK, FK	Arial, regular, 8 point, blue, mixed case
Text Block	Arial, regular font, 16 point, black, mixed case

### 5.5. Relationship Standards

- **Avoid crossing relationship lines.** Data models in which relationship lines do not cross are easier to read and appear less cluttered. Very large data models may have

trouble with this, especially in the “Main” subject area. But every effort should be made to minimize line crossings, especially in subject areas other than “Main”.

- **Never cross entities with relationship lines.** When a relationship line passes behind an entity that is not part of the relationship, it appears as if the entity actually is part of the relationship.
- **Stretch entities to avoid a cluttered effect from a significant number of relationships.** The more relationships in which an entity is involved, the larger the entity’s box needs to be to clearly show all of the incoming relationship lines.
- **Crow’s feet should point to the bottom or to the right.** This approach will place significant entities in the top left section of the diagram, and parent-child relationships will generally “read” from top to bottom and left to right. Although not always possible, every effort should be made to follow this rule, especially in subject areas other than “Main”.
- **Center subtype entities below the supertype entity.** I.e., place all of the subtype entities in a horizontal row below the supertype entity, with the supertype entity roughly in the middle of the horizontal row. This will make clear the supertype-subtype relationship. If the diagram makes this arrangement impossible, the next best alternative is to vertically arrange subtype entities to the right of the supertype entity, with the supertype entity roughly in the middle of the vertical row. Never arrange subtype entities on multiple sides of their supertype entity.

## 5.6. Invoking Standards within ERwin

Within the ERwin 4.1 tool, some aspects of naming and data type standards can be managed automatically by setting certain options. Only a select group of standards management tasks are identified below. For full information on the features, consult Erwin’s online help.

The following options can be found under the “Tools/Names/Model Naming Options”:

- Under the “General” tab, the naming standards and abbreviations file to be used at the Ohio Department of Education can be identified. It is required that all data models attach ODE’s global naming standards file through this interface. The names and abbreviations in this file are updated on a regular basis. The file to be used is named simply “ODE”, stored in the “Templates” library in ModelMart.
- Under the “Logical” and “Physical” tabs, the case and length of Logical and Physical objects may be managed. A length of zero indicates no maximum length. The following settings should be used:

Tab	Object Type	Case	Maximum Length
Logical	Entities	UPPER	0
	Attributes	Initial	0
	Relationships	Lower	0
Physical	Tables	UPPER	25
	Columns	Lower	30
	Relationships	Lower	30
	Indexes	UPPER	30

- Under the “Name Mapping” tab, the conversion process from Logical to Physical names is specified. Checking the “Use Glossary” check boxes enables the use of the ‘ODE’ abbreviations file to automatically convert logical names to ODE approved abbreviated names.

The following options can be found under the “Tools/Names/Edit Naming Standards”.

- Under the “Logical” and “Physical” tabs, the Word Type (Prime, Modifier 1, Modifier 2, Class) of each node of a Logical or Physical object name can be identified so that adherence to naming standards can be validated.
- Under the “Glossary” tab, the content of the ‘ODE’ abbreviations file is displayed using Word, Abbreviation, and Word Type values. It is here that new words and abbreviations are added. Only the ModelMart administrator can add new words and abbreviations to the glossary, but data modelers should be prepared to provide a list of words that need added to the glossary based on the following rules:
  - Business words up to five characters in length do not need to be abbreviated. Six-character words are borderline; abbreviate the word if an obvious abbreviation exists. Any word over six characters should be abbreviated.
  - An abbreviation should be from two to five characters in length. Abbreviations over five characters should be avoided whenever possible.
  - Different forms of a common root word should all have the same abbreviation. For example, “detail”, “details”, and “detailing” should all share a single abbreviation such as “det!”.
  - Once a word is entered in the glossary, its abbreviation **may not** be changed. All data models share the same global naming file, and changing an abbreviation could impact the physical model in every one of them!
  - Consider adding borderline length words to the glossary with the word itself as its own “abbreviation”. This will protect against another data modeler believing it is safe to abbreviate a word because it is not in the glossary, when the word is in fact used in other data models.
- Under the “Tools/Names/Check Standards Compliance” choice, adherence to the identified naming standards can be verified and corrected.

## 6. Logical Data Model Standards

### 6.1. General Logical Naming Standards

The following naming standards apply to all objects<sup>2</sup> in a logical data model.

- **A name must convey concise and accurate information about the object being defined.** It should describe the object precisely and be as self-documenting as possible. The name should be identifiable outside the context of the application system and should stand alone. For example, “Grade Code” is a poor attribute name because “grade” can convey different meanings depending on the context. Is this a test’s target grade level, a student’s current grade level, the inclination (grade) of a hillside, the quality of a cut of meat (grade “A”), etc. A better name might be “Proficiency Test Grade Code”, the meaning of which is clearer.
- **Words used in logical names should not be abbreviated.** For example, “Number” should be fully spelled out, not abbreviated to “Num”.
- **Acronyms may not be used.** Acronyms are essentially jargon and are not self-documenting, so even those that are well understood within ODE should be fully spelled out. For example, “Information Retrieval Number” should be used instead of “IRN”. Exceptions may be made on a case-by-case basis if the phrase appears so often in the model that it actually clutters the model, or if multiple acronyms present in the same name result in a *very* long name when fully spelled out (e.g. 100 characters). But when in doubt, always err on the side of verbosity.
- **A name should not be encoded, so that no specific knowledge of the business context is required.** For example, “Subcontractor Payment Code” is a better name choice than “Regulation 257 Code”. Using “Regulation 257 Code” requires someone to know that the internal company Regulation 257 deals with the types of subcontractor payments.
- **A name must be singular, not plural.** For example, LINE ITEM, not LINE ITEMS; and PERSON, not PEOPLE.
- **A single blank space is placed between each word in the logical name.** Avoid the use of special characters, such as dashes or underscores. The first character of a name must not be a blank space.
- **Avoid using articles, prepositions or conjunctions.** These words include (but are not limited to): a, an, the, and, but, or, as, after, because, if, when, in, through.
- **Avoid redundant references.** Do not repeat the meaning of a word or use more than one similar word to describe an object. For example, “Student Enrollment Count Quantity” is redundant because the words ‘count’ and ‘quantity’ convey similar meanings.

---

<sup>2</sup> The term “Object” refers to an item in a logical or multi-dimensional data model or physical database, such as: Entity, Attribute, Relationship, Table, Column, Index, Primary Key, Dimension, Member, etc.

- **There are exceptions to every rule.** Deviations from naming standards are sometimes necessary to accommodate certain business terminology or special requirements. Every attempt should be made to minimize this type of condition. All exemptions must be approved during the review process.

## 6.2. Logical Data Model Entity Standards

### 6.2.1. Entity Name

Entity names in a logical data model are subject to the general logical naming standards outlined in section 6.1. In addition, the following rules apply:

- **The name of an entity should convey a high level description of the contents (attributes) of the entity.** An entity should contain attributes that define a single concept, and the name should reflect that concept.
- **In a header/detail scenario, the child entity should be named after the parent entity.** The preferred name will be the parent entity's name appended with the words "LINE ITEM". For example, PURCHASE ORDER and PURCHASE ORDER LINE ITEM, or INVOICE and INVOICE LINE ITEM. This naming format should be followed unless it can be shown that "line item" does not make sense in a data model's specific context.
- **A "correlation" entity must never be created to resolve a many-to-many relationship.** The proper way to handle many-to-many relationships is to create the relationship as a many-to-many relationship in the first place. Erwin will create a "physical only" table in the physical model to resolve the many-to-many relationship. (Note that this only applies to many-to-many relationships between two entities. Three-way or more relationships must be manually modeled.)
- **REFERENCE Entities:** Entities created for code lookups should be named REFERENCE <lookup>, where <lookup> is the name of the code attribute that is the key of the lookup, minus the class word "Code". For example, the lookup entity for an attribute named Test Subject Code should be REFERENCE TEST SUBJECT.

### 6.2.2. Entity Definition

All entities must have a definition. Entity definitions in a logical data model are subject to the rules outlined in section 6.1. In addition, the following rules apply:

- **When referring to an entity, the entity name should be in all upper case letters.** This immediately tells the reader that the concept being discussed exists elsewhere in the data model. For example, PERSON ORGANIZATION ROLE is defined as "The distinct function or part played (i.e. duties and/or responsibilities for) by a PERSON within an ORGANIZATION."

- **For the definition of a code-type (REFERENCE) entity:** Use the construct “A list of valid codes representing ... as recognized by the Ohio Department of Education”. For example, the code table of district types is defined as “A list of valid codes representing the types of school districts recognized by the Ohio Department of Education.”
- **For status snapshot entities:** Use the construct “A historical record of the state of ...”. For example, REFERENCE APPROVAL STATUS SNAPSHOT is defined as “A historical record of the state of permission granted for a request to make a change to or add the entry for a new ORGANIZATION for recognition by the Ohio Department of Education.”

### 6.2.3. Entity Note

An Entity Note is a freeform text field used for additional temporary information about the entity such as modeling issues, questions, or status updates about the entity. Before a data model is promoted into the QA library these issues must all be resolved, and therefore this field must be empty before the data model can be promoted.

### 6.2.4. Entity User-Defined-Properties (UDPs)

UDPs are fields provided by ERwin to be used for the collection and storage of additional metadata, customized to meet the specific needs of the data model. A UDP provides the most effective method to collect and store this additional permanent metadata in a format that can be easily extracted from the model for interfacing with metadata processes.

The Ohio Department of Education has defined the following UDPs for use with entities. The use of UDPs is optional, except where mandated by the project sponsor.

- **Business Rule:** Business Rules that pertain to an entity that cannot be adequately documented in the model.
- **Alias Names:** Alternate names by which an entity may be known.
- **Stewardship:** Identifies the data steward, BSME, and/or TSME. Use a position or department name, not the name of an individual person. Individuals will change more frequently than departments and positions.
- **Source Doc:** A list of documents used a data source for the entity.
- **Logical Only Reason:** Documents the reason that an entity is logical only.

### 6.2.5. Entity Logical-Only Switch

A “check box” used to designate entities that are represented in the logical data model but not yet physically deployed. These are “placeholder” entities approved by the BSMEs that are expected to be physically deployed sometime in the future and may represent a “to be” business requirement instead of an “as is” requirement.

If logical only is checked, the Logical Only Reason UDP must be completed.

## 6.3. Logical Data Model Attribute Standards

### 6.3.1. Attribute Name

Attribute names in a logical data model are subject to the general logical naming standards outlined in section 6.1.

An attribute name is comprised of several words, with each word playing a role in building the meaning of the attribute name. At least two words must be present: a *prime* (business) *word* and a *class word* (domain). Additional *modifier words* are used when necessary to enhances meaning.

**Prime Word:** A significant business entity or concept within the enterprise, such as Organization, City, or Enrollment.

**Modifier Word:** Any descriptive word used to clarify the meaning of a name, such as First, Elapsed, Secondary, Status, Type, Delivery, Active, Default, or Completion.

**Class Word:** The name of the domain assigned to an attribute, such as Code, Description, or Date. One (and only one) class word is required for every attribute. The class word must match the domain assigned to the attribute, it may not be overridden. E.g., if the domain is KEY, then the class word must be “Key”; it cannot be changed to “Code” or “Identifier”. (See section 6.3.2 for more on domain assignments.)

The format of an attribute name is:

**Wwww Wwww Wwww Ccccc**

where:

- **Wwww** is either a prime or modifier word. At least one is required, and there is no maximum number allowed. Use as many modifier words as necessary to make the attribute name clear and self-documenting.
- **Ccccc** is the class word inherited from a domain.

Examples of attribute names are Organization Name, Status Begin Timestamp, Gender Code, and District Information Retrieval Number.

### 6.3.2. Attribute Domain

#### Domain Assignment

Each attribute name has certain metadata properties that describe it, such as a definition, data type, format, abbreviation, and allowed values. Within ERwin, a shortcut method exists to assign properties to an attribute using *domains*. Domain names map exactly to class words.

Assigning an ERwin domain enhances the self-documenting nature of an attribute name, because it provides immediate information about the attribute’s properties. For example,

if an attribute name is identified with a “Date” domain, some properties of the name are immediately known: it contains a month, a day, and a year.

Assigning a domain to each attribute, and including the domain name as the class word in the attribute name, is a mandatory task for the logical data modeler.

### **New Domains**

Only the ModelMart administrator is permitted to define new domains. If a data modeler has a suggestion for a new domain, it must be presented to the ModelMart administrator, who may suggest an alternate domain that is already defined. If it is decided that the suggested domain should be added to the domain dictionary, the following metadata about the new domain must be defined:

- **Domain Name:** The title of the domain. This name appears in all drop down lists used for Domain selection and management.
- **Domain Parent:** Categorizes domains by type. For example, the parent domain of “Count” is “Number”.
- **Domain Icon:** A graphical picture that visually represents the domain.
- **Domain Name inherited by Attribute:** A self-documenting word that provides immediate information about the attribute’s properties that is provided as the default name when an attribute/column is assigned to this domain. It must be appended to the end of each attribute name as a class word.
- **Domain Icon inherited by Attribute:** A graphical picture that visually represents the domain in data models where the stored display shows attribute detail. It is provided as the default icon when an attribute is assigned to a domain.
- **Domain Data Type:** A default logical data type and length for all attributes attached to this domain. A validation rule and default value may also be assigned.
- **Domain Definition:** The description of the domain.
- **Domain Definition inherited by Attribute:** The description of the Domain that is provided as the default description when an Attribute is assigned to this Domain. Include ellipses (...) to indicate where the data modeler must fill in the specifics for an attribute.

Before suggesting a new domain to the ModelMart administrator, make sure that the suggested domain follows a few simple guidelines.

- **It should have a specific and well understood meaning that does not overlap extensively with another existing domain.** For example, “Count” is an existing domain with a well understood meaning. Although “Headcount” is a term commonly used at ODE, it makes a poor domain candidate because “Count” already exists and can be used in place of “Headcount”.
- **It should be generic enough to use in many attributes of different meaning.** “Count” can be applied to any attribute involving a count of discrete elements. “Headcount” is limited in that it specifically implies a count of people.
- **It should be as atomic as possible.** “Fiscal Year” can be applied to several different types of attributes (State Fiscal Year, Federal Fiscal Year, etc.), but it is

not a good choice to be a domain because a more atomic domain exists within it. The domain “Year” can be used in attributes for fiscal years, as well as any other type of attribute containing a year value.

### 6.3.3. Attribute Definition

Attribute definitions in a logical data model are subject to the rules outlined in section 6.1. All attributes must have a definition.

It is recommended that a domain be assigned to an attribute before adding the attribute’s definition. All domains include a definition template that can be expanded to construct a complete attribute definition, potentially saving the data modeler some work. This is a guideline however, not a rule. A data modeler should write a definition from scratch when doing so results in a clearer definition than using the domain’s definition template.



**Tip:** The most common error made in creating attribute definitions is to violate the very first rule listed in section 6.1: *The terminology used in a definition must not be a restatement of the object name.* Metadata is very important at ODE, and this rule is rigorously enforced. If shortcuts are taken with definitions in an attempt to save time, the data model will likely be rejected during the review phase, eliminating any perceived time savings.

### 6.3.4. Attribute Note

An Attribute Note is a freeform text field used for additional temporary information about the attribute, such as modeling issues, questions, or status updates about the attribute. Before a data model is promoted into the QA library these issues must all be resolved, and therefore this field must be empty before the data model can be promoted.

### 6.3.5. Attribute User-Defined-Properties (UDPs)

UDPs are fields provided by ERwin to be used for the collection and storage of additional metadata, customized to meet the specific needs of the data model. A UDP provides the most effective method to collect and store this additional permanent metadata in a format that can be easily extracted from the model for interfacing with metadata processes.

The Ohio Department of Education has defined the following UDPs for use with attributes. The use of UDPs is optional, except where mandated by the project sponsor.

- **Business Rule:** Business rules that pertain to an attribute that cannot be adequately documented in the model.
- **Alias Names:** Alternate names by which an attribute may be known.
- **Source Doc:** A list of documents used a data source for the attribute.
- **Examples:** Example values for an attribute.
- **Logical Only Reason:** Documents the reason that an attribute is logical only.

### 6.3.6. Attribute Logical Only Flag

A “check box” used to designate attributes that are represented in the logical data model that may or may not be physically deployed in the future. These are “placeholder” attributes approved by the BSMEs that are expected to be physically deployed at some time in the future.

If logical only is checked, the Logical Only Reason UDP must be completed.

### 6.3.7. Reference Entity Attributes

For a code lookup entity named REFERENCE *<lookup>*, the entity should contain the following attributes:

<b><i>&lt;lookup&gt;</i> Code</b>	The code that is looked up. This is the primary key of the entity.
<b><i>&lt;lookup&gt;</i> Description</b>	The short description that defines the lookup code. If the “description” is actually a proper name, use <b><i>&lt;lookup&gt;</i> Name</b> instead.
<b><i>&lt;lookup&gt;</i> Text</b>	The long textual description that fully defines the lookup code. Optional.

If the nature of the code table is that codes have a limited life span but are not necessarily to be deleted when no longer needed, the following attributes should also be included:

<b><i>&lt;lookup&gt;</i> Start Date</b>	The first date that the code is/was valid for use.
<b><i>&lt;lookup&gt;</i> End Date</b>	The last date that the code was/will be valid for use.
<b><i>&lt;lookup&gt;</i> Current Flag</b>	Y/N flag indicating whether the code is currently valid for use.

## 6.4. LDM Relationship Standards

Relationships in a logical data model are subject to the general relationship standards described in section 5.5. The following sections define additional requirements specific to logical data models.

### 6.4.1. Verb Phrases and Relationship Names

A verb phrase plainly describes from a business perspective how the involved entities are related. A verb phrase must be created for every relationship.

The following rules govern relationship verb phrases:

- **A verb phrase must not use words that convey conditional states.** Words such as “may, always, could” are not permitted. That is automatically managed based on the cardinality and relationship type chosen for the relationship.
- **A verb phrase must take into account the “direction” of the relationship.** A verb phrase that may make sense from a parent-to-child direction may not make sense from a child-to-parent direction. When the full entity/verb phrase construct is read, it should flow like normal speech patterns with no awkwardness.

For example, between the ORGANIZATION and the child ORGANIZATION LOCATION, the verb phrase is: “resides at”, resulting in “ORGANIZATION resides at ORGANIZATION LOCATION”. Between child ORGANIZATION LOCATION and parent ORGANIZATION, the verb phrase is: “provides residency for”, resulting in “ORGANIZATION LOCATION provides residency for ORGANIZATION”.

- **Avoid excessive reuse of simple verb phrases.** Developing unique and appropriate relationship verb phrases can be challenging when many entities have similar relationships, but every attempt should be made so that the same phrase is not overused so that the meaning of a relationship is not diminished. For example, “defines”, “is”, “has” and “includes” are overused and overly simplistic relationship phrases.
- **Verb phrases for Parent-to-Child relationships are required.** Verb phrases for Child-to-Parent relationships are required only when the Child-to-Parent relationship is the more widely recognized relationship or is needed to clarify a business rule. When a Child-to-Parent relationship is not given a verb phrase, the default phrase generated by ERwin (of the form r/xxx) must be removed.
- **REFERENCE Entities:** For relationships between a parent REFERENCE (code) entity and a child entity, use a verb phrase such as “classifies”, “differentiates”, or “categorizes”. For example, “REFERENCE ORGANIZATION TYPE classifies ORGANIZATION”.
- **STATUS SNAPSHOT Entities:** For relationships between a parent entity and a child STATUS SNAPSHOT entity, use the standard phrase “has state recorded by”. For example, “ORGANIZATION has state recorded by APPROVAL STATUS SNAPSHOT”.

ERwin will generate a relationship name based on the names of the two related entities and one or more verb phrases. This name cannot be changed, other than by changing the entity names or the verb phrase.

The statement that is generated within the ERwin tool when a relationship name is derived is valuable metadata. It serves to validate the business rules governing the data model. The name of the Relationship should convey a high level description of how the involved parent and child entities interact with each other.

#### 6.4.2. Relationship Definition

Optional. Not used unless the relationship name is not self-explanatory.

#### 6.4.3. Role Name Information

When using an alias name as a substitute for the original attribute name, select the migrated attribute for which an alias is required, and then enter the alias name in the Role Name field.

#### 6.4.4. RI Actions

Unless specific considerations require them, the referential integrity default should be used. The defaults are:

	<b>Child</b>	<b>Parent</b>
<b>Delete</b>	None	Restrict
<b>Insert</b>	Restrict	None
<b>Update</b>	Restrict	Restrict

If RI is used, specify the appropriate action for delete, insert, and update of child and parent entities to describe the interrelationship between them.

## 7. Physical Data Model Standards

### 7.1. Physical Naming and Abbreviation Standards

Names for physical objects<sup>3</sup> are generally subject to restrictions imposed by the database management system (DBMS), e.g.:

- **Length of an object name.** Oracle imposes a maximum length of thirty characters to all object names.
- **Use of reserved words.** Certain words are used by a DBMS to manage its internal system objects and cannot be used in any other context. Reserved words may not be used on their own, but usually may be used as part of a longer name. E.g., the single word “system” cannot be an object name because it is a reserved word, but it is okay as part of the longer name “transportation\_system\_type”.
- **Blank spaces.** By default, blank spaces are not permitted in an object name.

These restrictions may make it impractical or impossible to use the unaltered full length logical names in a logical data model as physical object names in a physical data model and physical database.

ERwin includes the capability to manage naming standards and abbreviations. With appropriate parameters set within ERwin, the conversion of logical object names to physical object names is largely automated. This physical object name is created by looking up the logical object name’s component words and abbreviating them according to the glossary in the global naming standards file. (Note that some manual intervention may still be required when the physical name continues to exceed the RDBMS maximum name length after abbreviation.)

To enable this functionality, the global naming standards file must be attached to a data model. For more information on the naming standards file and abbreviations, see [section 5.6, “Invoking Standards within ERwin”](#).

### 7.2. Physical Data Model Table Standards

#### 7.2.1. Table Name

A default table name will be generated by ERwin by applying the naming standards file to the logical entity name. The generated name will consist of a series of abbreviated words connected by underscores (\_).

The generated name should not be changed unless the resulting name is longer than 25 characters. (Oracle supports 30 characters, but we need to reserve five characters for the prefixes used to construct index names, foreign key names, etc.) Manual additional

---

<sup>3</sup> A “physical object” can be a column, table, index, tablespace, or other DBMS component.

abbreviation and/or removal of words from the table name should be performed only to bring the table name length down to 25 characters or less.

Table names are always in uppercase characters. This is a documentation convention only; Oracle is not case sensitive.

### **7.2.2. Comment**

The comment will be copied from the logical entity definition. This should not generally be changed unless the physical implementation of the table differs from the concept documented by the logical entity. The table's comment will be stored in the database.

### **7.2.3. Volumetrics**

Values for Initial Rows, Max Rows, and Grow By \_\_\_ per Month should be entered if known at design time. Additionally the 'Average Width' and 'Percent Null' boxes should be filled out under the Oracle Tab of the Columns dialog box if known. This provides more accurate sizing estimates. (Note – ERwin assigns a zero avg. width to NUMBER datatypes). Accurate data for these items can be obtained through simple SQL statements where data currently exists in a relational table. Estimates are acceptable if exact figures are unavailable.

### **7.2.4. Physical Properties**

Specifying the tablespace in which the table will be created is required. There are usually multiple tablespaces created in each database, one of which is designated specifically for tables.

The remaining physical storage properties are not required. Tablespaces are created with AUTOALLOCATE functionality, meaning that Oracle automatically determines the amount of disk space to allocate to a table based on row size and table growth.

### **7.2.5. Partitions**

Fill out the partitioning information if a table is to be implemented as a partitioned table. Partitioning is unusual outside of data warehousing scenarios, but may sometimes be necessary to improve performance of particularly large tables in an OLTP environment (e.g. tables with millions of rows) or when data is to be retired in a "rolling window" fashion.

Name each partition *P\_name 1\_name2*, where:

- *name1* provides an indication of how the table is partitioned. This must be the same for all partitions within a table.
- *name2* identifies the specific range of values contained in a specific partition. This name is different for each partition.

For example, a table partitioned on a DATE column so that each partition contains data for one fiscal year might have partitions named P\_FISCAL\_YEAR\_2002, P\_FISCAL\_YEAR\_2003, etc. (Tip: Partition names must be unique only within a single table, not within the entire schema.)

## 7.3. Physical Data Model Column Standards

### 7.3.1. Column Name

A default column name will be generated by ERwin by applying the naming standards file to the logical attribute name. The generated name will consist of a series of abbreviated words connected by underscores (\_).

The maximum length of a column name in Oracle is 30 characters. In cases where the use of standard abbreviations results in a name that exceeds the applicable character limit, the column name is truncated to 30 characters. To resolve this, additional manual abbreviation action is required. Observe the following guidelines when performing manual abbreviation:

- An abbreviation should ideally begin with the same letter as the original full word. If the business word begins with a vowel, there may be deviations from this guideline in the interest of shortening names as much as possible.
- Column Name format is **aaaa\_aaaa\_aaaa\_ccc** where:
  - **aaaa** is the abbreviation for either a Business or Modifier Word found in the corresponding attribute name. At least one is required.
  - **ccc** is the class word abbreviation. This is required and should not be changed or further abbreviated.
  - A single underscore (\_) is placed between each abbreviation.

Examples of abbreviated column names are: cli\_exec\_spsr\_name, relo\_rqr\_flag, and empmt\_strt\_date.



**Note:** Except for the special case described in this section, do not override the physical name in the model that ERwin generates. Overriding the ERwin physical name has several negative effects:

- It creates a rogue abbreviation that does not exist in the naming standards file
- It unbinds the physical column name from the logical attribute name, preventing all automatic abbreviations in that column
- It obscures the long column name from being properly identified during review process, preventing the candidate name from being entered into the naming standards file.

Column names are always in lowercase characters. This is a documentation convention only; Oracle is not case sensitive.

### 7.3.2. Datatype and Size

A column's datatype and size are set to default values based on the domain assigned to the logical attribute. These can be adjusted according to the physical needs of the column, with some restrictions:

- **Do not change the physical datatype to conflict with the domain parent's base data type.** For example, the parent domain of KEY is NUMBER, so the physical datatype must be based on Oracle's NUMBER datatype. Assigning a scale is permitted; e.g. changing the datatype to NUMBER(6) is permitted. However, changing the datatype to CHAR, VARCHAR2, or DATE is not permitted. *Exception:* The CODE domain is based on CHAR but may be changed to NUMBER. ODE uses both numeric and character codes in its applications.
- **Use only DBMS native datatypes.** For example, although Oracle will translate ANSI datatypes such as DECIMAL, the proper Oracle datatype is NUMBER.

### 7.3.3. Comment

The column comment will be copied from the logical attribute definition. This should not generally be changed unless the physical implementation of the column differs from the concept documented by the logical attribute. The column's comment will be stored in the database.

## 7.4. Foreign Key Standards

Relationships in the logical data model become foreign key constraints in the physical data model. In the physical data model, the relationship name will not be used for the foreign key constraint name. The constraint name will instead be based on FK\_<table>\_<parent\_tbl>, where <table> is the table name (or an abbreviation) and <parent\_tbl> is the table being linked to (or an abbreviation). This approach will often require using abbreviations to meet the 30-character limit for object names.

To create or modify a foreign key constraint name, select a relationship between two tables. Right click on the relationship line and select relationship properties. On the "General" tab there is a field entitled 'Foreign Key Constraint Name'. Enter a constraint name as described above, performing any manual abbreviation necessary to keep the constraint name length within the 30 character limit.

## 7.5. Index Standards

### 7.5.1. Index Name

Index names should be of the format <id>nn\_<tbl\_name>, where:

- <id> represents the type of index. Use one of the following constants:

- **PK** – Primary Key
- **AK** – Alternate Key (Unique index)
- **IX** – Non-unique indexes (inversion entries), including indexes on foreign key columns
- **IB** – Bitmap indexes
- **nn** is an integer incremented for each index of the same type placed on a table. Primary Key indexes do not have an integer suffix because a table can have only one primary key.
- **<tbl\_name>** is the abbreviated table name

Examples: AK01\_ADDRESS, IX02\_PERSON\_LOCATION, PK\_ADDRESS



**Tip:** Erwin automatically creates index definitions for all foreign key columns, but by default they are hidden from view and flagged to not be generated. Instead of defining new indexes on foreign key columns, use these predefined indexes:

1. Check the “Show FK Indexes” box on the main index form.
2. For each FK index that should be created, click the “Generate” box on the index’s Oracle tab.
3. Rename the FK indexes to conform to ODE index naming standards.

### 7.5.2. Index Physical Properties

Specifying the tablespace in which the index will be created is required. There are usually multiple tablespaces created in each database, one of which is designated specifically for indexes.

The Unique check box must be selected if the index name begins with AK, indicating an alternate key. The Bitmap check box must be selected if the index name begins with IB, indicating a bitmap index. Both check boxes must be unchecked if the index name begins with IX, indicating a standard index.

The remaining physical storage properties are not required. Tablespaces are created with AUTOALLOCATE functionality, meaning that Oracle automatically determines the amount of disk space to allocate to a table based on row size and table growth. Other storage properties may be set if the data modeler determines a need to override Oracle’s automatic settings.

### 7.5.3. Index Partitioning

Fill out the partitioning information if an index is to be implemented as a partitioned index. Partitioning is unusual outside of data warehousing scenarios, but may sometimes be necessary to improve performance of particularly large tables in an OLTP environment (e.g. tables with millions of rows).

Partitioned indexes should only be created on partitioned tables. The index should be created with local partitioning, i.e. the partition keys and ranges should be identical for the index and underlying table. The same partition names should be used as well.

Some indexes may not be locally partitioned. E.g., a primary key index cannot be locally partitioned unless the partition key is one of the indexed columns. If local partitioning is not possible, a global (non-partitioned) index should be created.

## 7.6. Other Physical Model Object Standards

In most cases, the abbreviated table name is the starting point for creating other physical names for DBMS objects.

Database management systems (DBMSs) limit object names to various maximum character lengths based on the object type and other factors. The limit must be observed when specifying any applicable name.

Some object names are created by combining a prefix assignment (i.e., PKnn\_, IXnn\_, VWnn\_, etc.) and a descriptive base name. Because a prefix can add additional characters to the name, the base name may require manual abbreviation to meet the overall character limit for that object. Specific requirements are identified below.

Listed below is a sampling of the ORACLE DBMS objects for which naming standards must be manually applied. As requirements are identified for additional DBMS objects, the section will be updated to include the appropriate naming guidelines.

### 7.6.1. Check Constraint Standards

Check constraint names should be of the format **CKnn\_<tbl\_name>**, where:

- **<tbl\_name>** is the abbreviated table name, 25 character maximum
- **CK** is constant for “Check Constraint”
- **nn** is an incrementing integer for each check constraint placed on a given table

Examples: CK01\_ORG, CK02\_ORG

There are three exceptions to this rule. These exceptions exist due to the commonality of specific types of check constraints.

**Not Null constraints:** When a column in a CREATE TABLE statement is specified to be NOT NULL, Oracle implements a CHECK constraint on the column to enforce the “not null” rule. These automatically generated constraints are named SYS%\_Cxxx, where xxx is a system generated number. It is not necessary to rename “not null”

constraints generated with such names to conform to the `CKnn_<tbl_name>` format, although the data modeler is free to do so if desired.

**CODE and IRN domains:** An ErWin validation rule named “check\_field\_length” is assigned to the CODE and IRN domains. It ensures the required number of characters are stored in a column, i.e. no spaces or null characters. A constraint will be generated with the name ‘check\_field\_lenghtnn’, where *nn* is a sequence number that makes the constraint name unique.

**FLAG domain:** An Erwin validation rule named “check\_Y\_N” is assigned to the FLAG domain. It ensures that the value entered into a column is either a ‘Y’ or ‘N’. A constraint will be generated with the name ‘check\_Y\_Nnn’, where *nn* is a sequence number that makes the constraint name unique.

### 7.6.2. View Standards

View names should be of the format **VW\_<name>**, where:

- **VW\_** is constant for a logical view
- **<name>**: is the name given to represent the view.

As with table names, view names should indicate what data the view contains. Erwin cannot generate abbreviated view names automatically, so the data modeler must manually make sure the proper abbreviated words are being used.

Some possible scenarios and example view names are:

#### **Scenario 1: A view that prejoins multiple tables**

This type of view might be created to simplify a complex join for less sophisticated users or reporting tools. For example, a common query on the OEDS database is to find all ORGs related to other ORGs. This requires a query that joins ORG (as a child entity) to ORG\_TO\_ORG\_RELAT and then back to ORG again (as a parent entity). A suggested name for this view might then be `VW_ORG_RELAT_ORG`.

#### **Scenario 2: A security view**

A security view is used to limit access to data inside a table. For example, the PERSON table stores social security numbers, a data element which raises privacy issues and is therefore too sensitive to allow most users to see. A security view can be created to select all columns in the PERSON table *except* social security number, and the view might be named `VW_PERSON_NO_SSN` or `VW_PERSON_SECURE`. The Oracle accounts that should not have access to social security numbers would be granted SELECT privileges on the view, but not on the base table. (Going one step further, a private synonym could be created in that account that makes the user access the view with the name PERSON instead of the view name.)

### 7.6.3. Sequence Standards

Sequences are usually used to generate primary key values for a table. In this case, the sequence should be named **SEQ\_<table>**, where <table> is the name of the table for which the sequence will generate primary keys.

If the sequence will not be serving primary key values to a specific table, substitute a name that describes the sequence's purpose for <table>; e.g., SEQ\_RANDOM\_SEED.

### 7.6.4. Support Tables

Support tables provide application specific operational support in the physical database environment. The need for this type of table and the resulting naming standards associated with each support table will evolve as an application is developed.

Support tables should be named in a way makes obvious that fact that it is a support table. For example, the data marts in ODE utilize a support table named ETL\_INDEXES. The use of the "ETL" prefix implies that this table is involved in the ETL (extraction, transformation, and load) process that populates the data mart.

Support tables are usually physical only. They exist to serve the functional needs of a "back end" process, not to implement a logical business concept.

### 7.6.5. Database Links

The use of database links is permissible with the following restrictions:

- A database link name must be identical in development, QA, and production. The database link must point to a database in the same application tier, however.

For example, to create a database link to the OEDS-R database schema , the database link would be defined in each environment according to the following table:

Tier	Database Link Name	Oracle Service Name
Development	OEDSR	APPSDEV
QA	OEDSR	APPSQA
Production	OEDSR	APPSDB

It is prohibited to create a database link that links to a database in a different application tier. For example, a database link in a QA database may not point to a database in development or production.



**Exception:** An ODE database administrator may temporarily violate this rule to accomplish a specific task, such as an initial data migration of cleaned and verified data from a QA database into production. Such exceptions must be temporary and the DBA must remove the database link as soon as the specific task for which it was created is complete.

- The database link should connect to the read-only account of the target database. If there is a need to write data in the target database, another method should be found. Often this will involve creating ETL processes in Informatica, the agency's standard data movement and transformation tool.
- Because database links cannot be documented directly in an Erwin data model, their existence must be noted in a database UDP.
- Tables in remote databases often correspond to "logical only" entities in a data model. Synonyms should be created for these tables to make it appear as if the remote table is a local table. For example, a synonym ORG could be created for ORG@OEDSR. Remote tables not documented as "logical only" entities are not necessary to the local business function of the database and should not receive a synonym.

## **7.7. Documenting Logical to Physical Model Transitions**

The process of transitioning from a logical to physical model entails several considerations:

- Some attributes and entities from the logical model will not be realized in the physical data model.
- There will be some data unique to the physical model such as system tables and table columns indicating data lineage.
- Logical entities and attributes may be denormalized when moving from the logical to physical model to improve query performance.

### **7.7.1. Logical Only Model Objects**

Logical only model objects will not appear when viewing the physical data model. These objects may include:

- Entities
- Attributes
- Relationships

The ERwin AllFusion Modeling tool can be used to generate a report of all logical only modeling objects. The report will list all entities, attributes, and relationships with the logical only check box selected when in the logical version of the data model.

### **7.7.2. Physical Only Model Objects**

Physical only model objects appear in the physical data model but do not appear in the logical data model. These objects may include:

- Tables

- Columns
- Foreign key constraints

Physical only model objects are created by selecting the physical only check box when in the physical version of the data model. Physical only data model objects will not appear when viewing the logical version of the data model.

### **7.7.3. Mapping Logical to Physical Model Denormalization**

When transitioning the logical data model to a physical version there are changes that occur to support easier access to the data and to enhance query performance. These changes may result from taking constructs that are unique to the logical model (example supertype, subtype entities) or from denormalizing specific columns to improve performance.

Traditionally these changes have been documented through a manually created spreadsheet. The ERwin AllFusion Modeler tool automates this process through the use of the transforms concept.

The logical to physical transformation of the data model must be supported by invoking the transform wizard for the appropriate type of transformation. These transformations include:

- Rolling down two tables and create a single denormalized table
- Rolling up two tables and create a single denormalized table
- Resolving many-to-many relationships
- Rolling up subtype relationships
- Rolling down subtype relationships
- Creating an identifying relationship between a supertype entity and its subtype entities
- Horizontally partitioning tables
- Vertically partitioning tables
- Column denormalization

In the ERwin AllFusion Modeler, the Transform Options are displayed on the transform toolbar. The transform toolbar is found under VIEW/TOOLBARS/TRANSFORMS.

## 8. Other Physical Database Standards

### 8.1. DBMS Environment

#### 8.1.1. Platform

All new database development will be implemented in Oracle. ODE currently implements Oracle 10.2 running on HP-UX (Unix).

SQL Server continues to be supported by ODE for the purpose of maintaining and enhancing existing applications based on SQL Server. If and when these applications are scheduled for a complete reengineering, they should be migrated to Oracle. The long term goal is for Database Services to support a single enterprise database platform for its custom applications.

SQL Server will continue to be used where necessary to support Microsoft enterprise products, e.g. Sharepoint.

#### 8.1.2. Instances

Most application databases will be housed together in a common application instance, APPSDB. Very large or otherwise significant databases may be instead created in a dedicated Oracle instance. The final decision on where to generate a database will be made by Database Services based on several factors, including (but not limited to) the required physical storage, expected growth, number of users, and performance requirements.

All instances are created on three different servers, mapping directly to ODE's change control policy of having a development environment, a QA (test) environment, and a production environment. For an example application EXMPL, the network service names of the three instances will be:

- **EXMPLDEV:** The development instance. The application DBA or data modeler will have full access to create objects this instance.
- **EXMPLQA:** The QA (test) instance. Only an ODE DBA can generate the database in QA. The application DBA will have full DML privileges, but no DDL privileges.
- **EXMPLDB:** The production instance. Only an ODE DBA can generate the database in production. The application DBA will have full DML privileges, but no DDL privileges.

#### 8.1.3. Tablespaces

All database environments contain many tablespaces, two of which are accessible to data modelers: one for data (tables) and one for indexes. For an example application EXMPL, the tablespaces will be named EXMPL\_DATA and EXMPL\_INDEX.

In a physical data model, it is required that all tables be assigned to the DATA tablespace and all indexes be assigned to the INDEX tablespace on their respective physical properties page. The tablespaces will be set to automatically manage the storage of their objects, so setting the other physical storage parameters for a table or index is not required (Oracle will usually ignore them).

#### 8.1.4. Schemas

All environments will contain three schemas:

- **Administrator:** The account with the highest privileges and that “owns” all database objects. The schema shall be named after the application, e.g. OEDS. (Note: Previously this schema was known as the ADMIN schema, and would be named as such; e.g. OEDS\_ADMIN. For compatibility, these ADMIN schema names are not being retrofitted to remove the ADMIN suffix. New administrator schemas *should not* include the ADMIN suffix, however.)
- **\_APP:** The account that the application will use to logon to the database. The APP account has full DML privileges, but no DDL privileges. E.g. OEDS\_APP.
- **\_USER:** A read-only account usable by anyone. E.g., external users can run reports or external applications can link to the data. E.g. OEDS\_USER.

The application DBA (if any) will be given the password to the administrator schema in the development environment only. In the QA and production environments, only the ODE DBAs will have the administrator schema credentials. The application DBA will be given the APP and USER credentials for all three environments.

Application programmers will be provided credentials for the USER account, but never the administrator schema. Credentials for the APP account will be shared at the discretion of the Applications Manager.

## 8.2. Unmodeled Physical Objects

Some physical database objects may be created that are not part of a physical data model because Erwin does not recognize them or does not support them very well. The standards for these objects are described below.

### 8.2.1. Materialized Views (Snapshots)

Materialized views (also known as snapshots) can serve two purposes: to create a refreshable snapshot from another database in the local database (replication), or to create summary tables to improve query performance on very large tables through Oracle’s query rewrite mechanism (most common in data warehousing environments). ERwin does not provide a strong mechanism for modeling materialized views, however.

When creating a materialized view, apply the following standards:

- For a replication snapshot, name the materialized view MV\_APP\_xxxxxx, where APP indicates the external data source and xxxxxx indicates what the snapshot contains. E.g., a snapshot containing currently open buildings from the OEDS-R database might be named MV\_OEDS\_OPEN\_BLDG.

- For a summary materialized view, name the materialized view `MVnn_<tbl_name>`, where `nn` is a sequential number and `<tbl_name>` is the base table being summarized. This approach takes into account that summaries are often generated at multiple levels, and are not usually accessed directly; they are accessed transparently through query rewrite, so knowing the exact name of a specific summary table is not required. E.g., a set of summary tables on table `FACT_PROF_TEST` would be named `MV01_FACT_PROF_TEST`, `MV02_FACT_PROF_TEST`, etc.
- When a materialized view is created, Oracle creates a compound index on all columns in the base `SELECT` statement's `GROUP BY` clause. This index name is of the format `I$SNP_<mview_name>`. It is not necessary to rename this index according to ODE's index name standards. However, all indexes created by the database administrator must follow the same index name guidelines for indexes on tables, as defined in section 7.5.1.
- As with tables, specifying a tablespace for the materialized view and any indexes created on the materialized view is required.

### 8.2.2. Triggers

Triggers should be named `TRG_<tbl_name>_<action>`, where:

- `<tbl_name>` is the name of the table that fires the trigger. It may be necessary to slightly abbreviate or truncate the table name to create a trigger name that is 30 characters or less.
- `<action>` indicates what action fires the trigger, and can be one of:
  - **I** – Inserts
  - **U** – Updates
  - **D** – Deletes

If multiple triggers are created on the same table for the same action (i.e. one that fires per transaction and one that fires per row), append a number to the end of the trigger name to create a unique name.

Triggers should not be used to enforce business logic, they are only permitted to perform system level tasks such as fetching primary key values from a sequence or creating audit log entries. The source code for triggers should be stored in Team Foundation System. It is not necessary to include the PL/SQL code for triggers in the physical data model, but ERwin allows it.

The default triggers created by ERwin to enforce referential integrity should never be generated in the database. The Oracle kernel handles referential integrity without external intervention.

### 8.2.3. Stored Procedures and Packages

Database Services discourages the use of stored procedures. The ODE application architecture consists of several discrete layers, including a presentation layer, a business logic layer (.NET application code), and a data persistence layer (Oracle

databases). Maintaining the separation between those layers makes the architecture more robust and flexible. Storing application code in the database violates the separation of layers.

From a less academic perspective, using PL/SQL imposes the following “real” restrictions on application development:

- Application developers must become proficient in PL/SQL. The ODE application environment is based on .NET, however. Although .NET programmers can choose from a large pool of supported programming languages, PL/SQL is not among them. So developers must learn a language they will not frequently use, which leads to less skill development and potentially poor quality code.
- Scalability is reduced. Because ODE uses .NET application servers, PL/SQL code cannot be moved out of the database and into the application server. If the code is written in the .NET layer, it can be deployed to any .NET server as scalability issues require.
- Developers have a much stronger development environment for .NET in Visual Studio that they do for PL/SQL.
- A DBA from Database Services is required to create any object in QA or production environments, including stored code. This results in extra work for the DBAs and potential bottlenecks in getting code compiled. Application architects have more direct control over deploying code to the .NET application servers.

Despite these concerns, there will be times when stored procedures are used in applications. The standards for stored procedures are:

- **No standalone procedures or functions.** All stored code must be grouped into packages. The functions and procedures in each package should have related functionality. For example, one function might contain code that assists in ETL processing, another might contain all of the code that implements transaction logging, etc.
- **Package names should be standardized.** The name should be of the format **<app>\_<function>**, where:
  - **<app>** is an abbreviation for the application name. Use the same abbreviation in the network service name. For example, if the network name for a database is EXMPLDB, then **<app>** is EXMPL.
  - **<function>** describes the functionality of the package, e.g. ETL or TRIGGERS.

Example package names are therefore EXMPL\_ETL and EXMPL\_LOGGING.

- **Use PL/SQL, not Java or .NET.** Although Oracle supports Java stored procedures, ODE does not currently have enough programmers skilled in Java to adequately support Java-based applications. Programmers and DBAs trained in PL/SQL are more common.

Because Oracle now supports .NET stored procedures, using .NET seems like a natural fit for ODE. However, because ODE has no experience (or time to explore) the use of .NET stored procedures, and Oracle DBAs have no knowledge of .NET and would be unable to adequately support .NET stored procedures, this Oracle feature is not to be used.

- **Save all code as SQL scripts.** All code that creates packages should be saved as a script in a plain text file with a .SQL extension. What to do with the file depends on what the package does:
  - If the code is part of a larger application, it should be provided to the applications group just like all other application source code. The code will be stored in Team Foundation Server's source code repository.
  - If the code is database specific (e.g. it belongs to the DBAs only, or it is part of an ETL process, etc.) then the script should be turned over to Database Services for management.
- **Additional programming standards may apply.** The Applications group may require other programming standards beyond what Database Service requires. Examples are code review, procedure naming standards, procedure length, etc. Consult the Applications group regarding such standards.

## Appendix A: Domain Dictionary

A domain is required as the last word of an attribute name. The list below identifies each domain name, its definition, and default datatype. The non-abbreviated word is used in composing an attribute name and column name. The default datatype and length are suggested standard formats to be used in the physical model.

Domain Name	Domain Definition	Definition Inherited by Attribute/Column	Domain Datatype <sup>4</sup>	Icon
<unknown>	A new attribute may have an unknown domain when initially created, but must have a domain assigned before the logical data model is approved.	A Domain must be assigned to this Attribute/Column.	CHAR(1)	?
Address	A free form description used to identify the location (usually for mailing purposes) of a person or organization.	Information which uniquely designates a mailing location for ...	VARCHAR(60)	
Amount	A numeric value indicating a basic U.S. monetary unit of dollars and cents.	The monetary value of ...	NUMBER(12,2)	
Average Daily Membership	A measurement used to estimate enrollment. ADM is calculated by dividing the sum of all possible student attendance days (attendance plus absence) by the actual days in session.	A measurement used to estimate enrollment for .... ADM is calculated by dividing the sum of all possible student attendance days (attendance plus absence) by the actual days in session.	NUMBER	
Blob	Used to store very large objects such as pictures, images, graphics, and sound waves in character format.	A [type of binary data stored; JPG photo, MP3 sound, etc.] representing ....	BLOB	
Code	A set of encoded, alphanumeric values representing assigned meanings.	An encoded, alphanumeric value representing ...	CHAR(2)	
Count	A numeric representation of a group of objects; a total amount or number of things.	A numeric value representing a count of ...	NUMBER	
Date	The date on which an event occurs.	The date on which ... occurs.	DATE	
Date Number	A numeric value used to identify a unique occurrence of a day, date, month, quarter,	The date on which ... occurs, represented as a numeric string in the format [format	NUMBER(8)	

<sup>4</sup> See RDBMS Physical Standards for a detailed description of the datatypes.

Domain Name	Domain Definition	Definition Inherited by Attribute/Column	Domain Datatype <sup>4</sup>	Icon
	or year.	string, e.g. YYYYMMDD].		
Datetime	A combination of date and time, often used to record the exact moment when an activity took place.	A combination of date and time used to record the exact moment when ... took place.	DATE	
Description	A brief alphanumeric text string used to define a person, place, or thing.	A brief, alphanumeric text string used to define ...	VARCHAR(60)	
Flag	An alphanumeric "switch" with a maximum of two values representing the opposite of each other.	A value representing whether or not ...	CHAR(1)	
Identifier	An alphanumeric value with no implied meaning that uniquely defines a person, place, or thing.	A non-encoded, alphanumeric value that uniquely designates a ... within ....	VARCHAR(36)	
Information Retrieval Number	A numeric string that uniquely identifies an organization to the Ohio Dept. of Education.	A six character numeric string that uniquely identifies ...	CHAR(6)	
Key	A system generated numeric value with no implied meaning that uniquely defines a person, place, or thing.	A system generated numeric value with no implied meaning used to identify a unique occurrence of ...	NUMBER	
Measurement	A numeric value representing the dimensions, capacity, or amount of something ascertained by measuring.	A value representing the [dimensions, capacity, or amount] of ...	NUMBER(11,4)	
Name	An alphanumeric word or phrase used as the distinctive proper name of a person, place, or thing.	An alphanumeric word or phrase used as the distinctive proper name of ...	VARCHAR(60)	
Number	A value representing a generic piece of numeric data.	A numeric value representing ...	NUMBER	
Percent	A numeric value representing the ratio of one number divided by another.	A numeric value representing ... in relation to ...	NUMBER	
Quantity	A numeric representation of a sum of objects; a total amount or number of things.	A numeric value used to identify a quantity of ...	NUMBER	
Ratio	A numeric value representing a quantitative relationship.	A numeric value used to represent a quantitative relationship between ... and ...	NUMBER	

Domain Name	Domain Definition	Definition Inherited by Attribute/Column	Domain Datatype <sup>4</sup>	Icon
String	The default domain for alphanumeric content - must be replaced by a specific domain before it is considered complete.	A Domain must be assigned to this Attribute/Column.	VARCHAR(20)	
Telephone Area Code	A three digit numeric value assigned to a North American telephone number to designate a geographic location.	A value assigned to designate the area code for a telephone number assigned to ...	NUMBER(3)	
Telephone Code	A set of numbers representing other codes sometimes required to fully define a telephone number.	A set of numbers representing ... as part of a full telephone number.	NUMBER(2)	
Telephone Number	A set of numbers representing a North American telephone number.	A set of numbers representing telephone number assigned to ...	NUMBER(7)	
Text	An extended string of alphanumeric content used to fully describe a person, place, or thing.	An extended string of content used to fully describe ...	VARCHAR(255)	
Timestamp	A numeric representation of a specific point or moment when something occurred	A representation of the moment in time when ... occurs.	DATE	
Year	A four-digit numeric value representing a year.	The four-digit year on which ... occurs.	NUMBER(4)	

## Appendix B: Database Project Plan Template

The following project plan template defines all steps required by Database Services to meet Change Control standards, from the initial requirements gathering to database design to final implementation. This template should be incorporated into formal full project plans by the project manager.

### Phase: Requirements Gathering

Task ID	Task	Resource	Time Estimate	Dependencies	Notes
A1	Provide DB Standards Documents to developer	ODE DBA			
A2	Meeting w/ developer to ensure understanding of standards	ODE DBA, contracted DBA, data modeler, and/or PM	1 hr	A1	
A3	Signoff on understanding of and agreement to standards	Contractor PM		A2	<b>Milestone - Standards Agreement</b>

### Phase: Design

Task ID	Task	Resource	Time Estimate	Dependencies	Notes
B1	Design logical data model	contracted data modeler	highly variable		
B2	Logical data model review - preliminary	ODE DBA	2-8 hr	B1	Optional, but recommended
B3	Logical data model revisions - preliminary	contracted data modeler	2-8 hr	B2	
B4	Identify logical model words that need added to global abbreviations file, provide list to ODE DBA	contracted data modeler	1-2 hr	B1	
B5	Create abbreviations for provided word list and add them to the global file	ODE DBA	30 mi	B4	
B6	Create physical data model	contracted data modeler	highly variable	B5	
B7	Physical data model review - preliminary	ODE DBA	2-4 hr	B3, B6	Optional, but recommended
B8	Physical data model revisions - preliminary	contracted data modeler	2-8 hr	B7	

B9	Develop ETL process to prepopulate database with data / convert data from legacy system	contracted ETL designer	highly variable	B6 or B8	
----	---	-------------------------	-----------------	----------	--

**Phase: Development Environment**

Task ID	Task	Resource	Time Estimate	Dependencies	Notes
C1	Provide database sizing estimates to ODE DBAs	contracted DBA		None	
C2	Create new development Oracle instance	ODE DBA	1 hr	None	If applicable; some DBs will live in APPSDEV
C3	Create development schemas/accounts with appropriate privileges	ODE DBA	30 mi	C2	
C4	Provide access information (e.g. username, pwd, network naming file) to people who need it	ODE DBA		C3	<b>Milestone - ODE development database work complete</b>
C5	Generate database in development environment	contracted DBA	30-60 mi	B6, C4	
C6	Prepopulate development database with data / convert data from legacy system	contracted ETL designer	variable	C5, B9	
C7	Open development database to developers writing code	contracted DBA		C6	<b>Milestone - all development database work complete</b>

**Phase: Promote to QA**

Task ID	Task	Resource	Time Estimate	Dependencies	Notes
D1	Logical data model review - formal	ODE DBA	2-8 hr	B1	
D1a	DBA review of logical data model	ODE DBA		B1	
D1b	Provide entity/attribute definition list from Erwin to BSME	ODE DBA		D1a	Risk of delay in model approval if BSME does not respond in timely manner
D1c	BSME review and approval of entity and attribute definitions	BSMEs		D1b	
D2	Logical data model revisions - formal	contracted data modeler	2-8 hr	D1	D1-D2 repeats until logical data model is approved

D3	Logical data model signoff	ODE DBA		D1, D2	<b>Milestone - Logical data model acceptance</b>
D4	Identify logical model words that need added to global abbreviations file, provide list to ODE DBA	contracted data modeler	1-2 hr	D3	
D5	Create abbreviations for provided word list and add them to the global file	ODE DBA	30 mi	D4	
D6	Physical data model review - formal	ODE DBA	2-4 hr	D5	
D7	Physical data model revisions - formal	contracted data modeler	2-8 hr	D6	D6-D7 repeats until physical data model is approved
D8	Physical data model signoff	ODE DBA		D6, D7	<b>Milestone - Physical data model acceptance</b>
D9	Create new QA Oracle instance	ODE DBA	1 hr	None	If applicable; some DBs will live in APPSQA
D10	Create QA schemas/accounts with appropriate privileges	ODE DBA	30 mi	D9	
D11	Promote data model to QA library in ModelMart	ODE DBA	15 mi	D8	
D12	Generate database in QA environment from QA data model	ODE DBA	30-60 mi	D10	
D13	Provide code for any required stored procedures, packages, or triggers to ODE DBA	contracted DBA			
D14	Compile code for stored code in QA database	ODE DBA	15-30 mi	D13	
D15	Set up access to database objects	ODE DBA	15-30 mi	D12, D14	E.g. create synonyms, grant privileges, and analyze tables
D16	Provide access information (e.g. username, pwd, network naming file) to people who need it	ODE DBA		D15	<b>Milestone - ODE QA database work complete</b>
D17	Prepopulate QA database with data / convert data from legacy system	contracted ETL designer	variable	D16, C6	
D18	Open QA database for application testing	ODE DBA		D17	<b>Milestone - All QA database work complete</b>

**Phase: QA Testing**

Task ID	Task	Resource	Time Estimate	Dependencies	Notes
E1	Update logical data model as necessary	contracted DBA			
E2	Logical data model review - QA updates	ODE DBA		E1	E1-E2 repeats until logical data model is approved
E3	Signoff on changes to logical model	ODE DBA		E1, E2	
E4	Update physical model as necessary	contracted DBA		E1	
E5	Physical data model review - QA updates	ODE DBA		E3, E4	E4-E5 repeats until physical data model is approved
E6	Signoff on changes to physical model	ODE DBA		E4, E5	
E7	Generate changes in development database	contracted DBA		E4	
E8	Generate changes in QA database	ODE DBA		E6	E1-E8 repeats until QA process determines that no further database changes are required
E9	Identify and correct any problems with ETL to repopulate database / convert legacy data	contracted ETL designer, testers, TSMEs		D17	
E10	Certify that QA database (and application) is ready for production	Project TSMEs			<b>Milestone - QA certified to be ready for production</b>

**Phase: Promote to Production**

Task ID	Task	Resource	Time Estimate	Dependencies	Notes
F1	Compare QA database to QA model, verify that they remain identical	ODE DBA	1-2 hr	E10	
F2	Create new production Oracle instance	ODE DBA	1 hr		If applicable; some DBs will live in APPSDB
F3	Create production schemas/accounts with appropriate privileges	ODE DBA	30 mi	F2	
F4	Promote data model to production library in ModelMart	ODE DBA	15 mi	F1	

F5	Generate database in production environment from production data model	ODE DBA	30-60 mi	F4	
F6	Compile code for stored code in production database from source code in QA database	ODE DBA	15-30 mi	F5	
F7	Set up access to database objects	ODE DBA	15-30 mi	F5, F6	E.g. create synonyms, grant privileges, and analyze tables
F8	Provide access information (e.g. username, pwd, network naming file) to people who need it	ODE DBA		F7	<b>Milestone - ODE production database work complete</b>
F9	Prepopulate production database with data / convert data from legacy system	contracted ETL designer	variable	F8	
F10	Open production database to application team	ODE DBA		F9	<b>Milestone - All production database work complete</b>

## Appendix C: Logical Data Model Review Form

<b>Data Model:</b>		<b>Data Modeler:</b>	
<b>Library Name:</b>		<b>Data Modeler Phone:</b>	
<b>Data Modeler E-Mail:</b>		<b>Project Name:</b>	
<b>Data Modeler Dept:</b>		<b>Review Date:</b>	
<b>Reviewed By:</b>		<b>Reviewer Phone:</b>	
<b>Reviewer E-Mail:</b>		<b>Reviewer Position:</b>	
<b>Reviewer Dept:</b>		<b>Model Approval Status:</b>	
<b>Model Approved Date:</b>		<b>Model Manager Approval Signature:</b>	

All Model objects that are disapproved will be noted in the comment column of the review sheet. The logical data model will not be moved into the QA or production library, nor will the physical database be implemented in the QA or production environment, until all issues are resolved.

### Logical Model Level Review

<b>Criteria</b>	<b>Assessment (Approved/Disapproved)</b>	<b>Comment (If disapproved list specific Model Objects and define the error.)</b>
<b>Model Properties</b>		
Model Name Conforms to standards		
Model Properties General Box complete & correct		
Model Definition Box complete & correct		
Model Notation selected and conforms to standards		
Model Properties User Defined Properties complete and accurate		
Model Properties Defaults conform to standards		
Model Properties RI Defaults complete and conform to standards		
<b>Model Presentation</b>		
Correct Model Template Used		
Text Box used for Legend and Reflects Selections in the Model		
Appropriate naming		

<b>Criteria</b>	<b>Assessment</b> (Approved/Disapproved)	<b>Comment</b> (If disapproved list specific Model Objects and define the error.)
Standards File attached to Model		
<b>Diagram Presentation</b>		
Entity names reflect ODE presentation standards (Fonts etc.)		
Attribute Names reflect ODE presentation standards (Fonts etc.)		
Relationship Lines reflect ODE presentation standards (Fonts etc.)		
Display Primary Key Designator (PK), Foreign Key Designator, and Alternate Key Designator options are display options are switched on		
Display Inherited Attribute option has been switched on		
Display Attribute Icon option has been switched on		
<b>Relationships</b>		
Display Relationship Verb option has been switched on		
No Relationship Line Crossing behind entities		
Avoid Relationship Lines crossing other relationship lines. (Exception: Main Subject Area for larger models)		
All Relationships have a verb phrase		
Verb Phrases are meaningful and accurately reflect business relationships between entities		
Accurate cardinality assigned for parent and child involved in the relationship		

### Subject Area Level Review

Criteria	Assessment (Approved/Disapproved)	Comment (If disapproved list specific Model Objects and define the error.)
Subject Area Name Conforms to standards		
Subject Area Properties General Box complete & correct		
Subject Area Definition Box complete & correct		
All Members of the Subject Area conform to the business and or structural meaning of that define the subject area		
Reference Entities are listed in the REFERENCE Subject Area		
Model Contains all required Subject Areas		

### Entity Level Review

Criteria	Assessment (Approved/Disapproved)	Comment (If disapproved list specific Model Objects and define the error.)
Entity Name Conforms to standards		
Entity Definition complete, correct, and conforms to standards		
BSMEs and TSMEs have signed-off on each entity definition		
Entity Notes should be empty		
User Defined Properties completed where appropriate		
Entity Color corresponds to the data model color legend		
Logical Only Box selected where appropriate		
Entity represents one and only one concept		
Super-type/Sub-type constructs are valid		

**Attribute Level Review**

<b>Criteria</b>	<b>Assessment</b> (Approved/Disapproved)	<b>Comment</b> (If disapproved list specific Model Objects and define the error.)
Attribute Name Conforms to standards		
Attribute Definition complete, correct, and conforms to standards		
BSMEs and TSMEs have signed-off on each attribute definition		
A valid logical domain must be assigned to each attribute		
Validate the presence of at least one Prime Word and Class Word for the attribute		
Role Names have been used where appropriate		
Attribute Note should be empty		
User Defined Properties completed where appropriate		
Logical Only Box selected where appropriate		
The attribute represents one and only one fact about the entity		

## Appendix D: Physical Data Model Review Form

<b>Data Model:</b>		<b>Data Modeler:</b>	
<b>Library Name:</b>		<b>Data Modeler Phone:</b>	
<b>Data Modeler E-Mail:</b>		<b>Project Name:</b>	
<b>Data Modeler Dept:</b>		<b>Review Date:</b>	
<b>Reviewed By:</b>		<b>Reviewer Phone:</b>	
<b>Reviewer E-Mail:</b>		<b>Reviewer Position:</b>	
<b>Reviewer Dept:</b>		<b>Model Approval Status:</b>	
<b>Model Approved Date:</b>		<b>Model Manager Approval Signature:</b>	

All Model objects that are disapproved will be notated in the comment column of the review sheet. The physical data model will not be moved into the QA or production library nor will the physical database be implemented into the QA or production environments until all issues are resolved.

### Physical Model Level Review

<b>Criteria</b>	<b>Assessment</b> (Approved/Disapproved)	<b>Comment</b> (If disapproved list specific Model Objects and define the error.)
<b>Model Properties</b>		
Model Name Conforms to standards		
Model Properties General Box complete & correct		
Model Definition Box complete & correct		
Model Notation selected and conforms to standards		
Model Properties User Defined Properties complete and accurate		
Model Properties Defaults conform to standards		
Model Properties RI Defaults complete and conform to standards		
Referential Integrity Defaults are correctly set		
<b>Model Presentation</b>		
Correct Model Template Used		
Text Box used for Legend and Reflects Selections in the Model		

<b>Criteria</b>	<b>Assessment</b> (Approved/Disapproved)	<b>Comment</b> (If disapproved list specific Model Objects and define the error.)
Appropriate naming Standards File attached to Model		
Volumetrics parameters set for the physical data model volumetrics utility (under tools menu)		
<b>Diagram Presentation</b>		
Table names reflect ODE presentation standards (Fonts etc.)		
Column Names reflect ODE presentation standards (Fonts etc.)		
Relationship Lines reflect ODE presentation standards (Fonts etc.)		
Display View option has been switched on		
Display View Relation option has been switched on		
Display Primary Key Designator, Alternate Key Designator, and Foreign Key Designator display options are switched on		
<b>Relationships</b>		
Display Foreign Key constraint option display option is switched on		
No Relationship Line Crossing behind entities		
Avoid Relationship Lines crossing other relationship lines. (Exception: Main Subject Area for larger models)		
All Foreign Key constraints have been named according to ODE standards/ Requirements		
Accurate cardinality assigned for parent and child involved in the relationship		

<b>Criteria</b>	<b>Assessment</b> (Approved/Disapproved)	<b>Comment</b> (If disapproved list specific Model Objects and define the error.)
Mandatory and optional requirements documented for the parent and child in the relationship		

### Subject Area Level Review

<b>Criteria</b>	<b>Assessment</b> (Approved/Disapproved)	<b>Comment</b> (If disapproved list specific Model Objects and define the error.)
Subject Area Name Conforms to standards		
Subject Area Properties General Box complete & correct		
Subject Area Definition Box complete & correct		
All Members of the Subject Area conform to the business and or structural meaning of that define the subject area		
Reference Tables are listed in the REFERENCE Subject Area		
Model Contains all required Subject Areas		

### Table Level Review

<b>Criteria</b>	<b>Assessment</b> (Approved/Disapproved)	<b>Comment</b> (If disapproved list specific Model Objects and define the error.)
Table Name Conforms to standards		
Table Definition complete, correct, and conforms to standards		
Required physical storage parameters completed		
Table partitions defined according to standards		
User Defined Properties completed where appropriate		

<b>Criteria</b>	<b>Assessment</b> (Approved/Disapproved)	<b>Comment</b> (If disapproved list specific Model Objects and define the error.)
Table Color corresponds to the data model color legend		
Physical Only Box selected where appropriate		
Volumetrics sizing information entered		
Transforms fully documented for super-type sub-type denormalization		
Transforms fully documented for Table roll-ups and table roll-downs		
Transforms fully documented for all column denormalizations		
Horizontal and Vertical partitioning transforms documented where appropriate		
DBA and TSMEs have reviewed the denormalization		

### Column Level Review

<b>Criteria</b>	<b>Assessment</b> (Approved/Disapproved)	<b>Comment</b> (If disapproved list specific Model Objects and define the error.)
Column Name Conforms to standards		
Column Definition complete, correct, and conforms to standards		
Null option correctly assigned to the column		
A valid physical domain related to the logical domain must be assigned to each attribute		
User Defined Properties completed where appropriate		
Logical Only Box selected where appropriate		

**Other Objects Review**

<b>Criteria</b>	<b>Assessment</b> (Approved/Disapproved)	<b>Comment</b> (If disapproved list specific Model Objects and define the error.)
Indexes named according to standards		
Required physical storage parameters completed on indexes		
Check Constraints named according to standards		
Views named according to standards		
Sequences named according to standards		
Triggers named according to standards		
Packages named according to standards		
All stored procedures and/or functions grouped into packages		