|  |  |
| --- | --- |
| DEP Logo | **Department of Environmental Protection** |
| ***STD-09061805.2.0*** | ***Page 1 of 28*** |
| **Physical Data Modeling Standard** | |

# 

# Purpose

This document specifies the Florida Department of Environmental Protection’s (DEP) Physical Data Modeling Standard. The purpose of this standard is to ensure that DEP physical data models (database schema objects) have a consistent look and feel.

# Scope

This standard applies to all database schema development at DEP.

# Standard

1. All DEP database schemas shall follow Oracle database standards and guidelines found at the [***Oracle Technology Network***](http://www.oracle.com/technetwork/index.html)website.
2. Developers shall follow the ***Physical Data Modeling Specifications*** which is included as Appendix A to this standard. This specification provides technical guidelines, definitions, and references, including database naming standards, physical model diagram layout, and instructions for naming constraints, indexes, sequences, triggers and views. A list of the Oracle Reserved Words is included in Appendix B to this standard.
3. All DEP database schemas shall follow the DEP ***Database Object Coding Standard*** (STD-14121501).

# Deviation from Use

Any deviation from this standard must be approved by the Enterprise Application Services or the Portfolio Management Services Program Administrator and be documented in associated project documentation. The DEP Contract Manager shall also document and approve any deviations for contracted projects.

# Appendices Appendix A: *Physical Data Modeling Specifications*

Appendix B:*Oracle Reserved Words*

Appendix C: *Column Class Words*

# Approvals

**Approved by Warren Sponholtz, CIO** 4/21/2016

**Approval Date**

**Appendix A: Physical Data Modeling Specifications**

Table of Contents

[Purpose 1](#_Toc448409483)

[Scope 1](#_Toc448409484)

[Standard 1](#_Toc448409485)

[Deviation from Use 1](#_Toc448409486)

[Appendices Appendix A: *Physical Data Modeling Specifications* 1](#_Toc448409487)

[Approvals 1](#_Toc448409488)

[INTRODUCTION 5](#_Toc448409489)

[GRANDFATHER CLAUSE 5](#_Toc448409490)

[GENERAL STANDARDS 5](#_Toc448409491)

[DEP Supported Modeling Tool 5](#_Toc448409492)

[Location Data Standards 5](#_Toc448409493)

[TABLES 6](#_Toc448409494)

[Table Name 6](#_Toc448409495)

[Table Alias 7](#_Toc448409496)

[Table Comment 7](#_Toc448409497)

[TABLESPACES 8](#_Toc448409498)

[DATA Tablespace 8](#_Toc448409499)

[INDEX Tablespace 8](#_Toc448409500)

[LOB Tablespace 8](#_Toc448409501)

[GIS Tablespace 8](#_Toc448409502)

[COLUMNS 8](#_Toc448409503)

[Column Name 8](#_Toc448409504)

[Column Sequence in a Table 9](#_Toc448409505)

[Column Comment 9](#_Toc448409506)

[Required Columns 10](#_Toc448409507)

[CONSTRAINTS 11](#_Toc448409508)

[Primary Key 11](#_Toc448409509)

[Example Primary Key Constraint and Column Names 12](#_Toc448409510)

[Foreign Key 12](#_Toc448409511)

[Example Foreign Key Constraint and Column Names 13](#_Toc448409512)

[Unique Constraints 14](#_Toc448409513)

[Example Unique Key Constraint and Column Names 14](#_Toc448409514)

[Check Constraints 14](#_Toc448409515)

[Example Check Constraint and Column Names 15](#_Toc448409516)

[NOT NULL Constraints 15](#_Toc448409517)

[INDEXES 15](#_Toc448409518)

[Indexing of Columns with Less Than 15 Distinct Values 15](#_Toc448409519)

[Redundant Indexes 15](#_Toc448409520)

[SEQUENCES 15](#_Toc448409521)

[VIEWS 16](#_Toc448409522)

[View Comment 16](#_Toc448409523)

[TRIGGERS 16](#_Toc448409524)

[GRANTS 17](#_Toc448409525)

[MATERIALIZED VIEWS 17](#_Toc448409526)

[PHYSICAL MODEL DIAGRAM LAYOUT 18](#_Toc448409527)

[Object Placement 18](#_Toc448409528)

[Drawing Foreign Key Constraints 18](#_Toc448409529)

[Avoid Overcrowding 18](#_Toc448409530)

[Facilitate PMD Reading 18](#_Toc448409531)

[PMD Legend 19](#_Toc448409532)

[Oracle Reserved Words 22](#_Toc448409533)

[Column Class Words 24](#_Toc448409534)

# INTRODUCTION

The development team uses the business model to define the logical implementation of the system which is addressed in the Logical Data Modeling Standards. The logical implementation is transformed into a Physical Data Model (PDM) that represents the physical tables and other database objects (views, triggers, procedures, etc.), which will be accessed by the application.

Project teams may consult with the Database Administration (DBA) staff in the Office of Technology and Information Systems (OTIS) during the physical schema design. The OTIS DBA staff shall review and approve all PDMs.

This standard governs the expectations of the deliverable components that must be met for the PDM to pass the review. The PDM has three components:

1. The **Physical Schema Diagram**, which is the “picture of the Physical Data elements” (usually the tables and/or views) for the database.
2. The **Data Dictionary**, which consists of documentation of the tables, columns, and associated indexes and constraints.
3. The **SQL Data Definition Language** (DDL) for tables, indexes, constraints, views, triggers, procedures, packages, functions, and database links.

# GRANDFATHER CLAUSE

Existing legacy applications that are grandfathered with respect to changes in the standards will be brought into compliance over time as enhancement releases are fielded for a particular application. Maintenance releases are specifically waived with respect to a standards review.

# GENERAL STANDARDS

## DEP Supported Modeling Tool

Conceptual, Logical and Physical Data Models must be developed using the Oracle SQL Developer Data Modeler tool; both DEP and external development teams must use Oracle SQL Developer Data Modeler and the approved DEP code versioning repository – Subversion (SDI). The use of any other modeling tool/repository is not authorized.

## Location Data Standards

The established Location Data Standards will be adhered to for geospatial objects. Adherence to these supplementary standards is mandatory. Models including geospatial objects will be sent to Geographic Information Systems (GIS) for review.

# TABLES

The table definition specifies what type of data the table can store and specifies referential integrity constraints used to restrict inserts, updates and deletions to the table’s data. ***Tables*** in the PDM must implement the business objects originally identified in the business model and Logical Data Model (LDM).

## Table Name

A table name must be plural. The following rules apply to table names:

* A table name is mandatory.
* It cannot exceed 30 characters.
* It must be upper case letters.
* It should be made up of one to five real words; abbreviations may be used only if the name cannot fit within 30 characters when the Table Alias is prepended.
* Valid examples are COUNTRIES, PEOPLE, and LEGAL\_CASES.
* The table name includes underscores in place of spaces, for example, SITE\_AGENCIES.
* Table names are not prefixed with an application abbreviation.
* When the table name is only one word, it must not be an [Oracle Reserved Word](#AppendixB).
* Table names must follow the guidelines for use of class words as follows:
  + **\_CODES** – Code/Look Up table names must end in \_CODES. Examples of valid code table names are STATUS\_TYPE\_CODES, STATE\_CODES, and ELEVATION\_CODES.
  + **\_HS** – Archive/History table names must contain the singular version of the table name for which it archives data and end in \_HS. Examples of valid archive table names are ACTIVITY\_HS (archive for the ACTIVITIES table), CONTRACT\_HS (archive for the CONTRACTS table), WORK\_IN\_PROCESS\_STANZA\_HS (archive for the WORK\_IN\_PROCESS\_STANZAS table).
  + **\_RPT** – Report tables are primarily used in the data warehouse. Report table names must end in \_RPT.
  + **\_MV** – Materialized views are generally reserved for use in the data warehouse and follow a different naming standard that what is listed in this context. Materialized views created in the transactional instances must end in \_MV to differentiate them from the primary tables. Materialized views in the data warehouse are usually maintained by the Database Administration team; the MV names match the corresponding source table.
  + **TEMP\_** – Temporary tables should be used sparingly and must be removed after a designated period of time. These table names must begin with the TEMP\_ class word.
  + **\_PRC** – Processing table names are using for manipulating data before passing it into another table. These tables must end with the \_PRC class word.

## Table Alias

Table aliases are given to all tables in the DEP transactional database instances. The aliases allow for quick identification of the owning schema for a table. The following rules apply to table aliases:

* All tables residing in one of the DEP transactional database instances must have an alias.
* The alias must follow the rules established in the Logical Data Modeling Standard and be registered when the Logical Model is completed.
* The alias must not be an [Oracle Reserved Word](#AppendixB).
* If the short name(s) have not yet been registered, they must be unique among all Oracle table short names used at DEP. See the BIS\_LIB.SHORT\_NAMES table in the ORADEV database instance for current short name values or get the short name(s) from your OTIS technical contact.
* The table short name must be unique among all existing DEP enterprise Oracle short names. Guidelines for determining a valid short name are:
  + Currently reserved short names are stored in the ORADEV database instance, BIS\_LIB schema, SHORT\_NAMES table.
  + A candidate table short name is subject to change until reserved.  Since multiple Application Development Teams must reserve short names, to prevent problems during development, Database Administration recommends that you submit your candidate short names as soon as possible to the DBA section when they are finalized.
  + Once a candidate table short name is established, the Application Development Team must contact the Database Administration Section who will in turn record (reserve) these short names and confirm they have been reserved.  This process changes the short name from a candidate to reserved status.
  + If the one word table name is six characters or fewer, use the first three letters of the name as the short name.  If that short name is has already been reserved, add one letter of the word until it is unique.
  + If the one word table name consists of more than six characters, use the first three letters of the table name as the short name.  If that short name is not unique, add one letter of the word, up to the sixth letter, until it is unique.
  + If the table name is more than one word, use the first character of each word up to six characters.
  + In the event that a duplicate table short name exists after applying the above rules, suffix the short name with the number ‘1’.
  + In the event that a table short name is initially six characters and is not unique, remove the last character and replace with the number ‘1’.  Keep incrementing this number until there is no longer a conflict with an existing short name.

## Table Comment

You must enter a comment for each table in the PDM. Comments are created in the database with the COMMENT ON command.

# TABLESPACES

Each schema must have two tablespaces: 1) the DATA tablespace, and 2) the INDEX tablespace. After you submit a request for initial set up of the schema, a member of the OTIS DBA staff will submit a request to the Agency of State Technology (AST) to create these tablespaces (<SCHEMA>\_DATA and <SCHEMA>\_INDEX) in the Oracle database.

Additional tablespace names may be requested for storage of LOB or GIS index data.

## DATA Tablespace

The DATA tablespace is the tablespace where standard data must be stored. The naming convention for the DATA tablespace is <SCHEMA>\_DATA. Examples of valid DATA tablespace names are STCM\_DATA, PEAS\_DATA, and CRA\_DATA.

## INDEX Tablespace

The INDEX tablespace is the tablespace where the physical aspect of the index definition must be stored. The naming convention for the INDEX tablespace is <SCHEMA>\_INDEX. Examples of valid INDEX tablespace names are STCM\_INDEX, PEAS\_INDEX, and CRA\_INDEX.

## LOB Tablespace

The LOB tablespace is the tablespace that may be requested when a schema contains multiple CLOB or BLOB columns and it is beneficial to segregate that data. The naming convention for the LOB tablespace is <SCHEMA>\_LOB. Examples of valid LOB tablespace names are REMOT\_LOB, FAMAS\_LOB, and WIN\_LOB.

## GIS Tablespace

The GIS tablespace is the tablespace that may be requested when a schema contains multiple Oracle Spatial (SYS.SDO\_GEOMETRY) columns and it is beneficial to segregate that data. The naming convention for the GIS tablespace is <SCHEMA>\_GIS. Examples of valid GIS tablespace names are WIN\_GIS, FDM\_GIS, and BIS\_LIB\_GIS

# COLUMNS

Columns in the PDM are the transformed business elements/attributes originally identified in the business model and Logical Data Model (LDM). Columns must adhere to the following rules:

## Column Name

* Column names are mandatory.
* Underscores “\_” are used to separate words instead of blanks.
* Column names must not be prefixed with the application schema name.
* Names must be singular.
* The column name, when a single word, must not be an [Oracle Reserved Words](#AppendixB) as defined in Appendix B.
* Column names must follow the guidelines for use of [Class Words](#AppendixC) as found in Appendix C.
* The column data type must be determined by the business need as follows:
  + The DATE data type must be used to store dates that do not require time zone or down to the millisecond.
  + The TIMESTAMP data type may be used if time zone is required or the time must be stored to the millisecond.
  + The NUMBER data type must be qualified with a precision with one exception, if the business need requires that the column contain either positive or negative numbers of any precision then it may simply be declared as NUMBER.
  + The VARCHAR2 data type precision may be defined up to 32,000 characters.
* When a database sequence is used to derive the value for a Primary Key column, the preferred data type is NUMBER(10) unless there is a business requirement for a larger precision. For a larger precision, the data type must be declared with the precision. Examples of larger precisions are NUMBER(15) and NUMBER(20).

## Column Sequence in a Table

The following guidelines apply to the required column order within a table:

* The Primary Key (PK) attribute must be displayed first.
* The required Foreign Key (FK) columns, if they exist, must be displayed after the Primary Key column in logical order of importance.
* The audit columns CREATE USER NAME and CREATE TS must be displayed next and in the same order in all tables.
* The required business columns are displayed next in logical business order. For example, the columns for an address would be ordered as ADDRESS 1, ADDRESS 2, CITY, STATE, ZIP5, and ZIP4.
* The optional business columns are displayed next in logical business order.
* The optional audit columns, MODIFY USER NAME and MODIFY TS are displayed last and must be in the same order in all tables.

## Column Comment

You must enter a comment for each column in the PDM. Comments are created in the database with the COMMENT ON COLUMN command. The following rules apply to Column Comments:

* Comments are limited to a maximum of 4000 characters in length.
* Comments must clearly indicate the columns’ meaning within the context of the enterprise rather than focusing on a comment that only has meaning to the application expert user group.
* The comment must define the business meaning; as such the system analyst/end users are the best source for a comment that avoids techno-jargon.
* Comments must stand alone. A comment would not assume the reader has access to other documentation.
* All comments must assume that the reader does not have direct knowledge of the application, that is, they will be enterprise comments rather than narrowly focused application comments.
* Circular comments must be avoided. In other words they should provide information about the attribute that extends the business definition beyond what is implied by the attribute name.

## Required Columns

There are six different table types categorized for use in Application Development—Archive, Code, Data, Processing, Report and Temporary. Depending on the table type, certain columns may be required to ensure quality table design. The six table types that OTIS has recognized and any audit column recommendations are documented below:

| **TABLE TYPE DESCRIPTION** | **REQUIRED COLUMNS** |
| --- | --- |
| **Archive –** a table that contains historical or archive data. An archive table must be named using the \_HS convention. | No audit columns are required. |
| **Code –** a table that contains repeatedly used data values that fall within the management of the application administrative team. A code table often populates the code selection fields in applications and is used for data validation. A code table must be named using the \_CODES convention. | **CREATE\_USER\_NAME** VARCHAR2(30) NOT NULL |
| **CREATE\_TS** DATE NOT NULL |
| **BEGIN\_DATE** DATE NOT NULL |
| **END\_DATE** DATE |
| **MODIFY\_USER\_NAME** VARCHAR2(30) |
| **MODIFY\_TS** DATE |
| **Data –** a table that contains one of the application’s core data. The data tables are those to which the users input and/or retrieve data from on a regular basis. | **CREATE\_USER\_NAME** VARCHAR2(30) NOT NULL |
| **CREATE\_TS** DATE NOT NULL |
| **MODIFY\_USER\_NAME** VARCHAR2(30) |
| **MODIFY\_TS** DATE |
| **Processing –** a table used during a system process (batch job, upload, download, etc.) before moving the data into another permanent table. A processing table must be named using the \_PRC convention. | **DATA LAST UPDATED DATE** DATE |
| **Report –** A report table is a table that may contain denormalized data. A report table is also called a results table. The table must contain a column for storing the last data that data was refreshed or added to the table. A report table must be named using the \_RPT convention. | **DATA\_LOAD\_DATE** DATE NOT NULL |
| **Temporary –** A temporary table is a table that is only required for a short period or was created temporarily for testing. It is used and then deleted following a process or it has been created on a temporary basis for testing and will be deleted in a timely manner. A temporary table must be named using the TEMP\_ convention. | No audit columns are required. |

# CONSTRAINTS

This section addresses primary key, foreign key, unique key, check and NOT NULL constraints.

## Primary Key

On a Physical Schema Diagram, the primary key indicator (#) denotes a primary key definition. The primary key must be made up of only one column which is specified as part of the constraint definition. You must define a primary key constraint for most tables in a schema. A primary key ensures each table row is uniquely identified and no duplicate rows are entered in that table. Processing, audit, report and temporary tables are exempt from this primary key rule but should contain adequate indexing to ensure proper performance. Some important characteristics of primary key constraints are:

* The primary key data value must always be unique within the table.
* The column defined as the primary key must be mandatory (NOT NULL).
* Foreign keys in join tables may reference the primary key.
* The primary key column name must be derived from the table name. Abbreviations to fit within the 30 character limit are not allowed.
* Primary key columns must end in the class word **\_KEY**. In the majority of cases the column is a surrogate key that is populated by an Oracle sequence.
* Primary key columns with intelligent data, such as code values, also end in **\_KEY**. These columns do not require Oracle sequences to populate the data.
* Primary key constraints are named with the table short name concatenated with the suffix “PK”, e.g., FAC\_PK.
* Primary key constraints are implemented with an ALTER TABLE USING INDEX clause, as are unique key constraints, to generate the index with the appropriate name.

Each primary key must follow the rules below for formatting a Primary Key column name. For data tables using an Oracle sequence, the Primary Key column name is usually the singular table name with **\_KEY** added. For code tables and tables that do not use an Oracle sequence as the Primary Key column, the column name is usually the singular table name minus the word CODE with **\_KEY** appended.

# Example Primary Key Constraint and Column Names

| **#** | **Table name** | **Primary Key**  **Column name** | **Table Short Name** | **Primary Key**  **Constraint Name** |
| --- | --- | --- | --- | --- |
| 1 | PROJECTS | PROJECT\_KEY | PRO | PRO\_PK |
| 2 | OBJECT\_CODES  (code table with an surrogate key as the primary key) | OBJECT\_KEY | OC | OC\_PK |
| 3 | STATE\_CODES  (code table with an intelligent key as the primary key) | STATE\_KEY | SC | SC\_PK |
| 4 | PAYROLLS | PAYROLL\_KEY | PAY | PAY\_PK |
| 5 | ACTIVITY\_VIOLATIONS | ACTIVITY\_VIOLATION\_KEY | AV | AV\_PK |

## Foreign Key

A foreign key constraint must be named with the two short names from each end of the relationship, (the first short name is that of the table that will own the foreign key) concatenated with the suffix FK for Foreign Key (e.g., CF\_FAC\_FK), where CF is the short name for COUNTY\_FACILITIES and FAC is the short name for the FACILITIES table.

The following rules apply to foreign key column names:

* Foreign key names are mandatory.
* Foreign key column names must match the primary key column with the parent table short name pre-pended, they cannot be abbreviated to fit within the 30 character limit.
* The true foreign key column names are named using the following convention: <TABLE ALIAS> \_<PRIMARY KEY COLUMN NAME>, where the short name and primary key column name are for the table and column that are being referenced.
* Foreign key constraints must have corresponding indexes (non-unique) generated to enhance performance. Foreign key constraint names would be named using the following convention: <CHILD TABLE ALIAS>\_<PARENT TABLE ALIAS>\_FK.

Some examples of valid foreign key column names are PAY\_PAYROLL\_KEY, FAC\_FACILITY\_KEY, and PRO\_PROJECT\_KEY.

# Example Foreign Key Constraint and Column Names

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Short Name of table with PK** | **Primary Key**  **Column Name** | **Foreign Key Column Name** | **Short Name of table referencing PK** | **Foreign Key Constraint Name** |
| 1 | PRO | PROJECT\_KEY  (table name PROJECTS) | PRO\_PROJECT\_KEY | ATT | ATT\_PRO\_FK |
| 2 | OC | OBJECT\_KEY  (table name OBJECT\_CODES) | OC\_OBJECT\_KEY | PAY | PAY\_OC\_FK |
| 3 | PAY | PAYROLL\_KEY  (table name PAYROLLS) | PAY\_PAYROLL\_KEY | EMP | EMP\_PAY\_FK |
| 4 | OV | OUTPUT\_VALUE\_KEY  (table name OUTPUT\_VALUES) | OV\_OUTPUT\_VALUE\_KEY | CMNT | CMNT\_OV\_FK |
| 5 | OMC | OUTPUT\_MEASURE\_KEY  (table name OUTPUT\_MEASURES) | OMC\_OUTPUT\_MEASURE\_KEY | CMNT | CMNT\_OMC\_FK |
| 6 | WIP | WORK\_IN\_PROCESS\_STANZA\_KEY  (table name WORK\_IN\_PROCESS\_STANZAS) | WIP\_WORK\_IN\_PROCESS\_STANZA\_KEY | WIPH | WIPH\_WIP\_FK |

Sometimes two or more columns in one table relate to the same primary key in another table. When implemented, this results in two foreign key columns referencing the same primary key.

The following rules apply to foreign key column names when two within the same table relate to the same parent table:

* Foreign key columns must match the primary key column, they cannot be abbreviated.
* A business term must be appended to the end to differentiate the two foreign key columns.
* The foreign key column names would be named using the following convention: <TABLE ALIAS>\_<PRIMARY KEY COLUMN NAME>\_<BUSINESS TERM>, where the short name and primary key column name are for the table and column that are being referenced and the business term further identifies the column.
* Foreign key constraint names would be named using the following convention: <CHILD TABLE ALIAS>\_<PARENT TABLE ALIAS>\_FK\_<BUSINESS TERM>.

You may name the two foreign key columns according to the following example:

The two foreign key columns below are both referencing the primary key EMPLOYEE\_KEY in the EMPLOYEES table. (In the JOBS table, the columns are EMPLOYEE and EMPLOYEE\_BOSS respectively.)

| **#** | **Short Name of table with PK** | **Primary Key**  **Column name** | **Foreign Key column name** | **Short name of table referencing PK** | **Foreign Key Constraint Name** |
| --- | --- | --- | --- | --- | --- |
| 1 | EMP | EMPLOYEE\_KEY | EMP\_EMPLOYEE\_KEY | JOB | JOB\_EMP\_FK |
| 2 | EMP | EMPLOYEE\_KEY | EMP\_EMPLOYEE\_KEY\_BOSS | JOB | JOB\_EMP\_FK\_BOSS |

## Unique Constraints

A unique key constraint is used to specify that no two rows of a table can have duplicate values in a specified column or set of columns. A table may have zero, one, or many unique key constraints. The default unique constraint name must comply with the following rules for naming Unique Constraints:

* The name contains the table alias as a prefix.
* The alias is suffixed with \_UK.

# Example Unique Key Constraint and Column Names

| **#** | **Table name** | **Unique Key**  **Column name(s)** | **Table Short Name** | **Primary Key**  **Constraint Name** |
| --- | --- | --- | --- | --- |
| 1 | PROJECTS | PROJECT\_FILE\_TEXT | PRO | PRO\_UK |
| 2 | OBJECT\_CODES | MODULE\_ID | OC | OC\_UK |
| 3 | RELATED\_PARTIES | FIRST\_NAME  LAST\_NAME  BEGIN\_DATE | RP | RP\_UK |

## Check Constraints

The following rules apply to Check Constraints:

* Check constraint names are mandatory.
* If multiple check constraints are defined for a table, they must be suffixed with \_CK#.
* Each check constraint name must be prefixed with the table alias and an underscore, e.g. FNI\_CK.
* Check constraints cannot be used as the sole constraint for columns that contain more than 3 distinct data values that may be extended during future work. In this case the column should be created as a foreign key column related to a code table.
* Full definitions of the data values verified by the check constraint must be entered into the column comments.

# Example Check Constraint and Column Names

| **#** | **Table name** | **Column name** | **Table Short Name** | **Check**  **Constraint Name** |
| --- | --- | --- | --- | --- |
| 1 | PERFORMANCE\_ACTIVITY\_CODES | RESPONSIBILITY\_IND | PAC | PAC\_CK1 |
| 2 | PERFORMANCE\_ACTIVITY\_CODES | PASS\_THRU\_FUNDING\_INCLUDED\_IND | PAC | PAC\_CK2 |

## NOT NULL Constraints

If a column is identified as a mandatory column, the definition NOT NULL must be added to the column DDL. The NOT NULL constraint must be entered next to each constrained column in the CREATE TABLE definition. Primary Key and Unique Key columns must be defined as NOT NULL.

# INDEXES

The following rules apply to Indexes:

* Index names are mandatory.
* Each index name must begin with the table short name followed by an underscore‘\_’ and suffixed with \_I.
* Index names for general columns must begin with the table short name followed by an underscore ’\_’ followed by a short version of the column name, followed by an underscore ’\_’ and suffixed with \_I. e.g. CF\_BEGIN\_DATE\_I is an acceptable index name for the BEGIN\_DATE column in the CONTRACT\_FACILITIES table.
* An index associated with a foreign key constraint is named with the two short names at each end of the relationship, with the \_I suffix, e.g., CF\_FAC\_FK\_I.
* Table short names must be used to indicate ownership for the index, (e.g. EMP\_DEPT\_FK\_I is a foreign key index from the employees table to the departments table).

## Indexing of Columns with Less Than 15 Distinct Values

You do not need to place an index on a column that has less than 15 distinct data values. It requires at least one read (of one block) to get the index, and a second read to get the data. If the entire data set can be read into memory in two reads, there is no performance gain from using an index in this case.

## Redundant Indexes

Do not use redundant indexes. For example, if a composite index already exists for columns (col\_a, col\_b, col\_c), then queries on (col\_a) and (col\_a, col\_b) will use this index; there is no need to define an additional index on col\_a. However, this index will not be used with queries on (col\_a, col\_c), (col\_b), or (col\_c); in such situations, additional indexes may be required on col\_b and col\_c separately.

# SEQUENCES

A sequence number generator (database sequence) can be used to create automatically unique integer numbers for primary (surrogate) keys. The following rules apply to creating Sequences:

* Sequence names are mandatory.
* Sequences on a primary key must be named using the table short name (e.g. FAC\_SEQ).
* Each sequence name must include the suffix \_SEQ.
* If multiple sequences are required for a single table, the sequence name must be suffixed with \_SEQ# where the '#' increments sequentially. (Valid examples are CRA\_SEQ1, CRA\_SEQ2, STCM\_SEQ3, etc.)
* Set sequences to NOCACHE.
* Set to increment by 1.
* Set sequences to NOCYCLE.
* The sequence MAXVALUE must match the maximum for the data type. For example, if a column is NUMBER(10) the MAXVALUE for the sequence must be 9999999999.

# VIEWS

The following rules apply to Views:

* View names are mandatory.
* The view name is the singular form, with underscores in place of spaces (e.g. COUNTRY\_VW, PERSON\_VW, and SITE\_AGENCY\_VW) with \_VW affixed.
* A view may not be created that exactly mirrors a table unless there is a need to represent a table between two database instances.

## View Comment

You must enter a comment that includes the business definition for each view in the PDM. Comments are created in the database with the COMMENT ON command.

# TRIGGERS

Triggers contain an unnamed piece of internal trigger logic written in PL/SQL. This logic may call additional logic external to the trigger, e.g., a named PL/SQL procedure.

Triggers are similar to stored procedures, except that a trigger is automatically executed when data in the associated table are modified. The following rules apply regarding the requirements for Triggers:

* Trigger short names are mandatory.
* The naming standard is <Short table name>\_{B/A}{R/S}{I/U/D} WHERE
  + {B/A} stands for Before or After,
  + {R/S} stands for Row Level or Statement Level and
  + {I/U/D} stands for Insert, Update or Delete
* Example OA\_ARD represents the OA table alias name, with an After, Row Level, Delete trigger.
* Note examples of more naming conventions below.

Example: PS\_BRIU -- Before row trigger firing on insert or update of the PTR\_STAFFS table.

CREATE OR REPLACE TRIGGER ps\_briu

BEFORE INSERT OR UPDATE  
ON PTR\_STAFFS  
FOR EACH ROW  
DECLARE  
BEGIN

if inserting then

if :new.id is null then

select ps\_seq.nextval into :new.id from dual;

end if;  
 :new.who\_created := user;  
 :new.when\_created := sysdate;

elsif updating then

:new.who\_updated := user;  
 :new.when\_updated := sysdate;

end if;

END;

# GRANTS

Grants are placed on schema objects to secure the data and are generally given to accounts needing access to perform application functions. It is the database developer’s responsibility to identify the appropriate grants and apply them in the Physical model so that the appropriate grant script is placed on the database. The primary Oracle grants approved for use by the Database Administration section are:

* SELECT
* INSERT
* UPDATE
* DELETE
* EXECUTE

# MATERIALIZED VIEWS

Materialized Views may be documented in the Physical model but the Database Administration section is responsible for creating and maintaining the standard Materialized View scripts that are run on the data warehouse instance (DOPPLER). The database developer usually should not provide scripts for this object. An exception is granted when complex materialized views are required. In this case the database developer is responsible for creating the scripts and providing them to the Database Administration section.

When requesting materialized views it is the responsibility of the database developer to produce a script containing all SELECT grants for those views that need to be given for the application to run. The DBA section will transfer the grants from the single script into the appropriate script so that the grants are not lost if the materialized views have to be dropped and recreated.

# PHYSICAL MODEL DIAGRAM LAYOUT

## Object Placement

Each diagram must be well organized and easily understood. Do not try to put all of the information into one diagram unless it will fit and still be easy to read and understand. The following are rules for object placement on the PMD:

* Place fundamental objects (primary tables or views) to the top left of the diagram.
* Size objects so all columns can be seen and provide clear paths for all foreign keys.

## Drawing Foreign Key Constraints

The line from one table or view to another represents the foreign key between those tables or views. The following are rules regarding drawing foreign key constraints:

* Face crow’s feet to the right (East) or down (South). In other words, all of the “three point prongs” point to the right or down.
* Avoid line crossings, if possible.
* Avoid bent lines. (Straight lines are always preferred.)

## Avoid Overcrowding

You should limit the number of tables or views in each diagram so that the final diagram is not too crowded. Depending on the business need of the model, if possible a diagram should contain no more than 15 to 20 tables and/or views. If a single diagram would logically contain more than 15 to 20 objects, consider a logical grouping for the tables or views in one or more diagrams. For example, a legal case tracking system could be divided into two PMDs, one with all of the CASE related tables and/or views and the second with all ATTORNEYS related tables and/or views.

## Facilitate PMD Reading

To make the Physical Model Diagram easier to read and review:

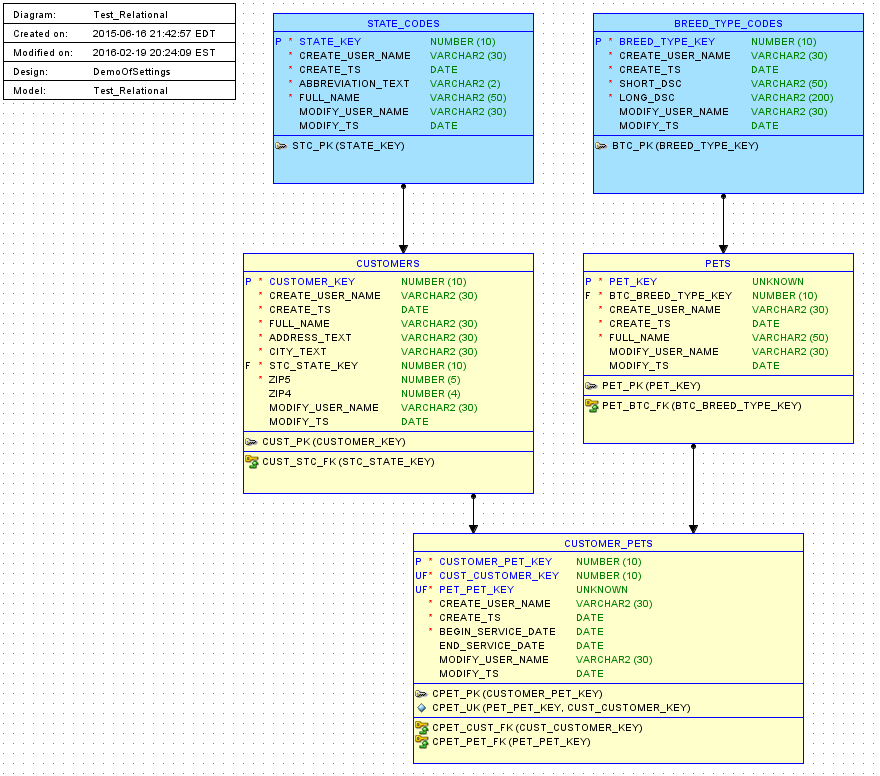
* Size objects so all columns can be seen and provide clear paths for all foreign keys.
* Make the font size of all table and/or view names at least 12 points, Bold. If you can use a larger font size, do so.
* For easier reading of the diagram:
  + Fill the tables and/or views with color to differentiate them from the lines representing the foreign key constraints.
  + Fill the tables and/or views of a certain type with the same color. See the table below for the current DEP color-coding standards for four identified table (or view) types.

| **Table or View Type** | **Fill Color** |
| --- | --- |
| **Data –** **A data table or view**. A data table (or view) contains the application’s core data. The table will be one to which the users input and/or retrieve data from on a regular basis. | **pale yellow** |
| **Code –** **A code table or view.** The table or view will contain repeatedly used data values that fall within the management of the application administrative team. A code table often populates the LOV fields in Oracle forms and is used for data validation. A code table should be named using the “ CODES” convention, for example “PAY\_CODES”. | **light blue** |
| **Archive –An archive table.** This type of table contains historical or archive data. | **pale green** |
| **Processing – An audit, report, or temporary table.** The table will contain data stored for audit purposes or that is reloaded daily, weekly, or on demand for some business function. | **light peach** |

## PMD Legend

A legend on the Physical Model Diagram is required to uniquely identify that diagram and to include the version date and time. Place the legend on the top, left corner of the diagram. You must include the following elements in the legend:

* Diagram Name
* Application Title
* Created Date
* Last Modified Date
* Author



**Appendix B: Oracle Reserved Words**

# Oracle Reserved Words

| **A-G** | **F-N** | **N-S** | **S-Z** |
| --- | --- | --- | --- |
| ACCESS | FILE | NOWAIT | SYNONYM |
| ADD | FLOAT | NULL | SYSDATE |
| ALL | FOR | NUMBER | TABLE |
| ALTER | FROM | OF | THEN |
| AND | GRANT | OFFLINE | TO |
| ANY | GROUP | ON | TRIGGER |
| AS | HAVING | ONLINE | UID |
| ASC | IDENTIFIED | OPTION | UNION |
| AUDIT | IMMEDIATE | OR | UNIQUE |
| BETWEEN | IN | ORDER | UPDATE |
| BY | INCREMENT | PCTFREE | USER |
| CHAR | INDEX | PRIOR | VALIDATE |
| CHECK | INITIAL | PRIVILEGES | VALUES |
| CLUSTER | INSERT | PUBLIC | VARCHAR |
| COLUMN | INTEGER | RAW | VARCHAR2 |
| COMMENT | INTERSECT | RENAME | VIEW |
| COMPRESS | INTO | RESOURCE | WHENEVER |
| CONNECT | IS | REVOKE | WHERE |
| CREATE | LEVEL | ROW | WITH |
| CURRENT | LIKE | ROWID |  |
| DATE | LOCK | ROWNUM |  |
| DECIMAL | LONG | ROWS |  |
| DEFAULT | MAXEXTENTS | SELECT |  |
| DELETE | MINUS | SESSION |  |
| DESC | MLSLABEL | SET |  |
| DISTINCT | MODE | SHARE |  |
| DROP | MODIFY | SIZE |  |
| ELSE | NOAUDIT | SMALLINT |  |
| EXCLUSIVE | NOCOMPRESS | START |  |
| EXISTS | NOT | SUCCESSFUL |  |

**Appendix C: Column Class Words**

# Column Class Words

The following authorized class words and their corresponding abbreviations are to be used as a suffix at the end of all column names. Each Class Word has one or more examples where the naming convention is applied in the Physical Model. The abbreviation for the Class Word will be used in the Physical Model to accommodate the attribute name length limitation, which is a maximum of 30 characters. Class word abbreviations not found in this list must be authorized by the Database Administration and Data Modeling Support section prior to use.

| **CLASS WORD** | **CLASS SUFFIX** | **DATA TYPE** | **DEFINITION** | **CORRECT USAGE EXAMPLES** | **INCORRECT USAGE EXAMPLES** |
| --- | --- | --- | --- | --- | --- |
| **Amount** | **AMT** | number | Applies to any monetary expression (i.e., dollars and/or cents). When this class word is used (with the AMT class suffix), it is suggested that a number format with a scale of 2 is used. The use of qualifier words is highly recommended to remove ambiguity. | * TAX EXEMPT AMT * WITHHOLDING AMT * DEDUCTIBLE AMT * BENEFIT AMT * AMT | * AMT TAX EXEMPT * WITHHOLDING AMOUNT * DEDUCTIBLE AMOUNT * AMT OF BENEFIT * AMOUNT |
| **Count** | **CNT** | number | The number or quantity of anything other than monetary amounts. The capture of a running total or summation from the actions of a specific event or set of occurrences of interest to the business. A calculated length of time must be expressed as a count of the appropriate units. The use of qualifier words is highly encouraged to remove ambiguity. | * WORK DAY HOUR CNT * DRIVER CONVICTION CNT * SPILL EVENT CNT * BILLABLE HOUR CNT * CNT | * WORK DAY HOUR COUNT * DRIVER CONVICTION COUNT * CNT SPILL EVENT * BILLABLE HOUR COUNT * COUNT |
| **Date** | **DATE** | date | A calendar date that will always contain the full century. This class word can never be used by itself, “DATE” is an Oracle reserved word. Date may also include the Hour Minute Second but not the Millisecond or Time Zone. If the Millisecond or Time Zone is required then use the TS class word. | * ACTIVITY ITEM BEGIN DATE * POLICY EFFECTIVE DATE * BEGIN DATE * END DATE * EFFECTIVE DATE * EXPIRATION DATE | * ACTIVITY ITEM BEGIN DT * DATE POLICY EFFECTIVE * BEGIN DT * EFFECTIVE DT * DT OF EXPIRATION * DATE |
| **Description** | **DSC** | varchar2 | A text string describing a business item that may have one or more sentences. It is end user designated and is used when most natural from a business point of view. The use of qualifier words is highly recommended to remove ambiguity. The class suffix should not be used by itself, the class word should be used for the column name in this case. | * SUBJECT DSC * INCIDENT DSC * COMPLAINT DSC * INCIDENT DSC * DESCRIPTION * DESCRIPTION | * SUBJECT DESCRIPTION * DESC OF INCIDENT * COMPLAINT DESCRIPTION * INCIDENT DESCRIPTION * DESC * DSC |
| **Geometry** | **GEO** | sdo\_ geometry | A set of data values that captures the spatial characteristics of an object. Note: The use of this class word will require the inclusion of additional attributes. Please refer to the GIS Location Data Standards (available on the GIS Standards (STD-09061811.1.1 at <https://floridadep-public.sharepoint.com/it-standards-library>) for these columns. The Oracle data type is always SDO\_GEOMETRY. | * INCIDENT LOCATION GEO * LAND GEO * WAFR GEO * WAFR OLD GEO * CHAZ GEO * LCTN GEO | * GEO INCIDENT LOCATION * LAND GEOMETRY * WAFR GEOMETRY * WAFR OLD GMTRY * CHAZ GEOMETRY * GEOMETRY |
| **Identifier** | **ID** | number varchar2 | A unique value (numeric or alphanumeric) assigned to an entity to uniquely identify that entity. This class suffix is used for columns that are used in Unique Constraints but are not the primary key column. The abbreviated class word suffix **must not** be used alone (ID); this suffix requires the use of one or more qualifier words. | * AFFILIATION ID * APPLICATION ID * INDIVIDUAL ID * LAND TYPE ID * USER ID | * AFFILIATION IDENTIFIER * ID FOR APPLICATION * INDIVIDUAL IDENTIFIER * LAND TYPE IDENTIFIER * USER IDENTIFIER * IDENTIFIER |
| **Indicator** | **IND** | number varchar2 | A one character value that identifies the existence of a specific state or condition that has only two possible values or states, such N or Y. If NULL or SPACE is used as a possible value then this class word should NOT be used since there would be three possible values for the field. The use of qualifier words is highly recommended to remove ambiguity. | * EMP OVERTIME IND * FACILITY CONTAM IND * HAZARDOUS IND * LAND USE IND * OPS IND * CITIZEN IND * IND | * EMP OVERTIME INDICATOR * IND FACILITY CONTAM * INDICATOR HAZARDOUS * LAND USE INDICATOR * OPS INDICATOR * CITIZEN INDICATOR * INDICATOR |
| **Key** | **KEY** | number | Used to define the primary key column of a table. This column may be a system generated unique sequence number that acts as a surrogate key assigned to an entity to uniquely identify it or a column containing a unique data value. A surrogate key implies that the key assignment is arbitrary and has no business meaning. Please note that surrogate keys are guaranteed to be unique, but are not necessarily consecutive. The Key class word is generally used when the value is not “visible” to the business user and is required for database activity. The class word should not be used by itself. | * COMMENT KEY * ADDRESS KEY * INCIDENT KEY * REMARK KEY * REMARK\_KEY | * COMMENT KY * ADDRESS KY NBR * INCIDENT KY SEQ * REMARK SURROGATE NBR * KEY |
| **Name** | **NAME** | varchar2 | The primary means of referring to a person, place or thing. It may be end user designated and is used when most natural from a business point of view. The use of qualifier words is highly encouraged to reduce ambiguity. | * FIRST NAME * LAST NAME * PRIOR NAME * DIVISION NAME * POSITION TITLE NAME * NAME | * FIRST NM * LAST NM * NAME PRIOR * NAME OF DIVISION * POSITION TITLE NM * NM or NME |
| **Number** | **NBR** | number varchar2 | A unique value (numeric, alphanumeric, or combination) assigned to an entity occurrence to uniquely identify it. This class should only be used when the associated data element is widely known and accepted with the suffix of "Number". The use of qualifier words is urged to increase clarity. The class suffix should not be used by itself. | * CONTRACT NBR * EMPLOYEE NBR * FILE NBR * DIVISION NBR * DRIVER LICENSE NBR * PROCESSING NBR * TOTAL\_NBR | * NUM OF CONTRACT * EMPLOYEE NUMBER * FILE NO * DIVISION NUM * DRIVER LICENSE NO * PROCESSING NUMBER * NUMBER |
| **Percent** | **PCT** | number | A ratio between two data values (with number data types) expressed as a percentage. The use of qualifier words is suggested to remove ambiguity. The class suffix should not be used by itself. | * COMPLETE PCT * COMPLIANCE PCT * SALARY COMMISSION PCT * SURCHARGE PCT * WATER VAPOR PCT * TOTAL\_PCT | * COMPLETE PERCENTAGE * COMPLIANCE PERCENT * PCT SALARY COMMISSION * PCT SURCHARGE * WATER VAPOR PERCENT * PERCENT or PERCENTAGE |
| **Rate** | **RATE** | number | A value, which usually remains constant over time, which can be used for calculation or computational purposes. The use of qualifier words is strongly recommended to remove ambiguity. The class suffix should not be used by itself. | * ANNUAL USAGE RATE * DIRECT LABOR RATE * FIXED RATE * FLOW RATE * OVERHEAD RATE * TAX RATE * TOTAL\_RATE | * ANNUAL USAGE RT * RATE DIRECT LABOR * FIXED RT * FLOW RT * RATE OF OVERHEAD * TAX RT * RT |
| **Text** | **TEXT** | varchar2 | A group of sentences that captures knowledge about a business event or situation that uses a narrative style. Typically these attributes are free style text, contain more than one sentence, and provide an alternative class for Descriptions where this class is most natural from a business point of view. The use of qualifier words is highly encouraged to remove uncertainty. The class word should not be used by itself. | * COMMENT TEXT * ENFORCE ACTION TEXT * ERROR TEXT * FULL TEXT * EVENT NARRATIVE TEXT * INCIDENT TEXT * MESSAGE TEXT * DOCUMENT\_TEXT | * COMMENT TXT * ENFORCE ACTION TX * TEXT FOR ERROR * FULL TXT * TEXT EVENT NARRATIVE * TEXT OF INCIDENT * TEXT MESSAGE * TXT or TX |
| **Time** | **TIME** | timestamp | A specific point within a 24 hour period at which an event has or will occur. Use the TIMESTAMP data type to keep the most precise time, including time zone. If the portion of importance is the time, use the Time class word. If the date *and* exact time is relevant, use the Timestamp class. In order to remove ambiguity, use qualifier words. The class word should not be used by itself. | * ACTIVITY START TIME * OPERATING TIME * START TIME * STORE CLOSE TIME * STORE OPEN TIME * CURRENT\_TIME | * ACTIVITY START TM * OPERATING TM * TIME OF START * TIME STORE CLOSE * TM STORE OPEN * TM |
| **Timestamp** | **TS** | timestamp | The combination of date and time used to capture a specific date and point in time. This class is primarily used for audit column purposes. The abbreviated class word suffix **must not** be used by itself and requires the use of one or more qualifier words. | * CREATE TS * END TS * ENTRY TS * LAST UPDATE TS * MODIFY TS * INCIDENT TS | * CREATE TIMESTAMP * END TIME STAMP * TS ENTRY * LAST UPDATE TIMESTAMP * MODIFIED TIME STAMP * TIME STAMP FOR INCIDENT * TS or TIMESTAMP |