PL/SQL STARTER FRAMEWORK	Oracle Naming Standard

Author	Bill Coulam
Created Date	November, 1997
Version/Date	4.1.1, July 23 2009

Overview

Other than a brief section on conceptual and logical model naming, this document covers the naming conventions used when designing the objects within a physical model targeted for the Oracle database.

NOTE: Database items created outside of the database, such as control files, database files, and [s]pfiles are part of the Core DBA domain and will not be addressed here.

To some, conformance to a naming standard seems inconsequential. However, a solid naming scheme has been proven to heavily reduce maintenance and enhancement costs. A good standard also prevents namespace collisions, enables automation, and provides the ability to quickly recognize important attributes and context for a given object.

Read this document, then print out the Nutshell page and tape it to your wall. Look to it frequently until these standards become second nature.

Typographical Conventions

When studying the syntax given for each convention, it is useful to recognize the following symbols and their meaning:

Convention	Meaning
{item**}**	Curly braces indicate a required item
[item**]**	Square brackets indicate an optional item
|	The pipe symbol indicates choice. Choose between items separated by delimiting pipes.

Table of Contents

Naming Scheme in a Nutshell

Refer to the following formats when naming physical objects in the data model. Remember that many of the name components are optional

Object Type	Naming Scheme	Examples
Table	Normal: [System_] [Logical Group_] {Table/View Name} [_Table Type] Associative: [System_] {First Table Name/Code_Second Table Name/Code} Rule: Table names must be 26 characters or less	SMS_BATCH_HIST SMS_STATE_LICENSE_TYPE
View	[System_] [Logical Group_] {Table/View Name} [_View Type]	SMS_CURRENCY_VW
Table Partition	Range: P {YYYY} [_MM] [_Interval Identifier] List: [P] {Range | Generic List Partition Name}	P2008_05 A_M, N_Z or P1, P2
Column	{Name} [_Column Type] Rule: Column names should be 20 characters or less	PATRON_STATUS_CD

Constraint	PK/UK: {Table Name_} {PK|UK[#]}
	FK: {Child Table Code_} {Column Descriptor | Parent Table Code} _FK
	Check: {Table Code_} {Column Name | TBL} _{NN|CK[#]}
		SMS_UNIT_PK, UNIT_STATUS_UK
		SMS_UNIT_STATUS_ID_FK
		SMS_ACTIVE_FLG_CK

Index	{Table Code_} {Column Descriptor_} {[Index Type Indicator(s)]IX}[#]	GL_ACCT_CLOSE_DT_IX, PMT_HIST_COMP_UIX
Synonym	{Same Name} (of object the synonym is abstracting)	*Same as underlying object*
Sequence	{Table Name} _SQ	SMS_UNIT_SQ
Type	[System_] [Logical Group_] {Type Name} [_Type Modifier] T	SMS_CUST_ATTR_T, CUST_ADDR_TT
Queue	[System_] [Logical Group_] {Queue Name_} {Q | EQ}	CORE_EMAIL_Q
Package	[System_] [Logical Group_] {Purpose} [_Function Code][_PKG]	MONITOR_DB_ERR
Routine	[Action_] {Purpose}	GET_LAST_EFF_DT, LOAD_NEW_LDGR
Trigger	Normal: {Table Code_} [Purpose_] {B|A [I|U|D] [R]|S} [#] Instead-Of: {View Name} _TRG System: {D|U} {_Triggering Event} _TRG	TERM_BIUD, ACCT_AUDS ACCOUNTS_VW_TRG U_AFTER_LOGON_TRG
Tablespace	{System_} [Logical Group_] {Tablespace Type} [_##]	SMS_ARCH_DATA, SMS_INDEX
DB Link	{Link Purpose} [_LINK]	NREL, IRS_LINK

Guiding Principles

This naming scheme, a summary of the following document, is based upon the following principles:

- Be consistent (but do not perpetuate error)
- Balance clarity with brevity.
- Related things should group closely.
- A good name is self-documenting and does not mislead.

Oracle Naming Restrictions

Oracle limitations dictate that names:

- Must start with a letter.
- Are limited to 30 characters long.
- May only contain letters, digits 0-9, # and _
- Should avoid duplicating any Oracle keywords and reserved words.
- Are case insensitive, so use underscores instead of CamelCase to separate name tokens.

Note: *Avoid using "#" and "$" in object names as they often cause issues within other technologies that see these symbols as comment and substitution characters.*

Name Component Explanation

Note the components of the naming scheme formats, like System code, Logical Group code, etc. Some are optional in small data models, but critical in large models. These components provide additional context about an object within a complex schema. They are an aid to creating order from the chaos, grouping objects within the same subsystem, feature set, or functional area. They also serve as buffers preventing namespace collisions among systems that reside in the same schema or database, or that collaborate through database links.

System Code

The system code is a simple two to four character acronym for the department, product or system to which the data object belongs. The system code is required in schemas where multiple subsystems reside. It is not recommended to pack the tables for disparate systems into a single schema. It is a best practice to reserve one schema for each system that requires database tables.

Logical Group Code

In large schemas tables tend to “clump” together, where three or more logically related tables all describe a particular area of the model. It is a good practice to use the optional logical group code to assign a common prefix to each of the related tables, such as CUST for customer tables, ORD for ordering tables, and GL for general ledger tables.

Table Code

Until Oracle lifts the severe 30-character limitation, constraint and index names will be the least automatable part of the naming standard, since each require several name components to identify source, target and type. There just isn’t room for two full table names and a column name within 30 characters.

For these objects, a short abbreviation derived from the table name is required. Generally, the abbreviation is formed by using the first letter from each token in the table name. When duplicate abbreviations occur, add a character or two to distinguish one from the other.

Example: ORDER_HIST table code would be OH, CUST_PRIM_LANG would be CPL.

Type Code

There are many types of Oracle objects. Furthermore, within each type of object, there are different sorts of each. There are four types of views, four types of constraints, and many types of tables and indexes. Therefore, it is best practice to attach a short suffix that visually distinguishes the object type or purpose. These are further detailed within each section below.

Conceptual Model Naming

Entity Class

Use the singular. Describe the real-world class (or type) of something, as opposed to an instance of that class.

Avoid abbreviations, unless widely understood by the business.

Attribute

Name format of conceptual attributes is:

{Prime Entity} {Modifier(s)} {Attribute Class/Type}

Examples: Vendor Contract ID, Service Contact Type

Where the Prime Entity is the entity to which the attribute belongs.

The Modifiers are the words used for additional refinement, qualification, readability.

The Attribute Class/Type is the word representing the data type or “domain” of the attribute. For example Customer Account Number is a well-named attribute.

Avoid abbreviations, unless widely understood by the business.

Relationship

Describe the role that each entity plays in the relationship.

Where more than two entities are involved, treat the relationship as an entity class.

Table and View

A table or view name must be 26 characters or less, and is derived from the following components:

[System_] [Logical Group_] {Table/View Name} [_Table Type]

The only required component is the table/view name itself. If there is a need for further clarification or categorization as described earlier, use system, logical group and type/function codes wisely.

Table and view names are singular. Do not use the plural form.

The underscore character "_" should separate all word tokens.

Use the full entity name in the physical name where possible, but if a word token is over 8 characters long, it is strongly encouraged to abbreviate now, during initial design. For example, if the model has several entities with "Responsible" in the name, start today by abbreviating with "RSPB". If a name, fully abbreviated, still exceeds the 26 character rule, create a new compound abbreviation. For example, if numerous tables will be prefaced with AUTO_PROVISION, abbreviate now to AUTOPVN or ATPROV or AP as a new compound abbreviation.

NOTE: Refer to the accompanying Abbreviation and Acronym spreadsheet for common short tokens.

Here is a sample list of table/view type codes for use as suffixes:

Tables	
Type	**Description**
AUD	Tables that track who changed what and when.
GT	A global temporary table. Often used for staging incoming data.
LOAD	Tables used by ETL processes to record metadata about each load.
LOG	Tables dedicated to receiving log messages from application processes.
HIST	Tables used to offload old data from production tables, but keep it online. Often referred to as historical or archive tables.
STG	Interim tables meant to be filled and emptied during processing. These have fallen out of favor with the introduction of global temporary tables.
SUM	Summarized or rolled-up data tables.
[M|S]QT	Use Q for queue tables. Optionally add M or S to distinquish between multi-consumer and single-consumer queue tables.
TYPE	Reference tables that contain unique codes, IDs and names.
Views	
VW	Standard view.
MV	Materialized view.
UV	Updateable view. Use only when you want to state that this view was created expressly for updating multiple tables through the view as a single interface.
OV	Object view (one which uses Oracle's object-relational features)
Data Warehousing	
DIM	Dimension table
BRDG	Bridge table
ADIM	Junk Dimension (collection of fact attributes (flags, etc.) that don't have their own dimension table because of their low cardinality characteristics.
HH	Horizontal Hierarchy table
VH	Vertical Hierarchy table

XREF	Cross reference table (for fact-less facts with only two dimensions)
FACT	Fact table.
AGG	Aggregation fact table.
CON	Consistent Aggregation fact table.
DW	Staging table for data warehousing.

Lookup Table

Also known as parent, master, LOV, or reference tables, these tables assume the name of the data they contain. No special conventions are required but lookup tables often end with "_TYPE".

Associative Table

[System_] {First Table Name/Code_Second Table Name/Code}

Associative tables—also known as Cross Reference or Intersection tables—are used to resolve many-to-many relationships. Attempt to include the names of the two tables being joined. If this yields a name longer than 26 characters, abbreviate the names or use table codes instead.

Attributive Table

These tables are a result of the first rule of normalization: No repeating attributes. No special conventions are required for these types of tables. The name is usually formed from the name of the parent table plus the repeating attribute, e.g ORDER as parent and ORDER_ITEM_DTL as the attributive child.

Subtype Table

These tables are also known as subset or sub-entity tables. These tables are foreign-keyed to their parent in a one-to-one relationship and usually contain attributes specific to a certain type of instance of the data represented in the parent table.

No special conventions are required for naming these tables, although use of the parent table name is encouraged. This will place the closely related tables next to one another in the object listings.

Table Partition

For standard date range table partitions, a partition name has the following components:

Range: P {YYYY} [_MM] [_Interval Identifier]

MM stands for the month number. The additional interval identifier is 1-5 for weekly partitions, or 01-31 for daily partitions. The "P" literal prefix is necessary since partition names can't start with a number.

Monthly partitions	P2003_05, P2007_01
Weekly partitions	P2003_05_2, P2007_01_4
Daily partitions	P2003_05_02, P2007_01_04, P2007_02_28

List partition names use the following components:

List: [P] {Range | Generic List Partition Name}

List partition names require a name derived from the values bounded by the range of that partition, like A_G, H_N, O_T, U_Z for alphabetical partitions on cities beginning with those letters. However, if the list partition range is likely to shift over time, it is better to give each list partition a generic name, like P1, P2, P3, etc.

Column

Column names should be 20 characters or less, and use the following components:

{Name}[_Column Type Code]

The 20 character goal is meant to leave enough characters free for the prefixes and suffixes required by the rest of the naming scheme (indexes and constraints in particular).

The underscore character "_" should separate all word tokens.

If the column is used in a foreign key to a parent table, the column name must match the name of parent key column.

Use the full attribute name in the physical name where possible, but if a word token is over 8 characters long, it is strongly encouraged to abbreviate now, during initial design.

NOTE: Refer to the accompanying Abbreviation and Acronym spreadsheet for common short tokens.

If the new column belongs to one of the following common column domains, it should be suffixed by the column type code.

Domain Type Code	Domain Type Spec	Description
NUM	INTEGER or VARCHAR2(#)	Number columns have intrinsic meaning. They are assigned, derived or computed. If letters, leading zeroes or special characters, like dashes, parenthesis, or dots must be preserved, use VARCHAR2 as the datatype and constrain to the max length expected. Examples are license numbers, social security numbers, & telephone numbers.
ID	INTEGER NOT NULL	Identifier. A unique numeric identifier with no intrinsic meaning. IDs are usually generated by a sequence and used as values for surrogate keys.
GUID	VARCHAR2(32)	Globally Unique Identifier. Do not use the SYS_GUID function of Oracle to produce GUIDs for surrogate keys. This column type is reserved for front end code to persist GUIDs generated outside the database.
BY	VARCHAR2(100)	Used for columns that preserve the user name or ID of the person who initiated a given action, as in AUTH_BY, CREATE_BY and MOD_BY.
CD	VARCHAR2(20)	Code. Short alphanumeric code, most often used as the primary or secondary unique key of a reference table.
NM	VARCHAR2(255)	Name. Used for longer alphanumeric identifiers that are not codes.
SNM	VARCHAR2(30)	Short Name. Short version of a name, often desired for UI controls, like drop-down lists, and reporting purposes to conserve screen/paper real estate.
DSCR	VARCHAR2(500)	Description. More informative than code or name, a short description of the attribute or entity instance.
NOTE	VARCHAR2(4000)	Free-form, typically user-entered, text about the instance of the entity or attribute.

YN	VARCHAR2(1) DEFAULT 'N' NOT NULL	Yes/No Flag. Values Y (true) and N (false) are the only valid values. NULL must not be allowed as a value.	
FLG	NUMBER(1,0) DEFAULT 0 NOT NULL	Numeric flag. Values 1 (true) and 0 (false) are the only valid values. NULL must not be allowed as a value.	
DT	DATE	Date. Avoid DTH or DTM (indicated truncation to hour or inclusion of time). Things tend to change and date precision shifts, so the generic DT is preferred.	
TS TSZ TSLZ	TIMESTAMP [(#)] [WITH [LOCAL] TIME ZONE]	Timestamps are used where sub-second precision is required. Use TSZ if timezone is included, TSLZ if the system is distributed and localized timestamps are needed. The number of fractional seconds should be dictated by requirements, but the default of 6 is typically sufficient.	
CT	INTEGER	Count.	Counts
QTY		Quantity.	
TOT		Total.	
AMT	NUMBER(X,Y)	Amount.	Monetary
RATE		Rate.	
PRICE			
COST			
PCT	NUMBER(X,Y)	Percent. Stored in decimal form. Consumers of percent columns multiply by 100 and round 0-2 places to get the user-friendly percentage.	Measurements
HT		Height.	
WT		Weight.	
LEN		Length.	

Constraint

Primary and Unique Key

Key constraint names are composed of only two components, therefore lending themselves well to automation:

{Table Name_} {PK|UK[#]}

Composing a primary key is as easy as adding PK as the suffix to the table name. Unique keys are equally easy to construct unless the table has more than one unique key, in which case the optional increment is used, as in APP_CODE_UK and APP_CODE_UK2.

NOTE:	Unless found on a reference table, more than one unique key on a table usually indicates an error in the data model.

Foreign Key

A foreign key name is composed of:

{Child Table Code_} {Column Descriptor | Parent Table Code} _FK

For single-column foreign keys, the Column Descriptor is the column name. For multi-column foreign keys, use the parent table code instead of truncating or abbreviating column names in the attempt to fit within Oracle's 30 character limitation.

In the event that the Table Code + the Column Descriptor + the suffix yields a name longer than 30 characters, the column name will need be compressed or truncated to fit.

Check

A user-defined check constraint name is composed of:

{Table Code_} {Column Name | TBL} _{NN|CK[#]}

For a column-level check constraint, the column descriptor is the column name: For example: LCN_LAT_NUM_CK.

For a table level check constraint, the use the table name. For example: VOTER_GENDER_CK.

Sequentially number the constraints if there is more than one table-level check, or more than one check on the same column (not recommended). For example: LDR_TYPE_CK and LDR_TYPE_CK2.

Not Null

System-generated NOT NULL check constraints will have a different name on every database, rendering false positives in tools that compare data models across environments for consistency. For this reason consider naming every NOT NULL check constraint. Use the "_NN" suffix for these types of check constraints.

Index

An index name is made from the following components:

{Table Code_} {Column Descriptor_} {[Index Type Indicator(s)] IX} [#]

If the index supports a unique constraint, the format is much more simple, mirroring the constraint name:

{Table Name_} {PK | UK}

See the next section for guidelines on naming Oracle Text indexes and accompanying preferences.

For a single-column index, the column descriptor is the column name. If space is limited after applying the required prefix and suffix, compress the column name by abbreviation and truncation.

For a composite index, the column descriptor is “COMP". When there is more than one such index on a table, the column descriptors for subsequent indexes are sequentially numbered (RT_COMP_IX, RT_COMP_IX2, etc.)

The optional, but recommended, index type indicators represent various index features. The following shows the index types in the order in which they should be concatenated together with the IX suffix.

Index Type Indicator	**Category**	**Explanation**
L	Partitioning	Index locally partitioned (equi-partitioned with the underlying table’s partitioning key)
G	Partitioning	Index globally partitioned (using column(s) that does not match the underlying table’s partitioning key)
B	Feature	Bitmap
F	Feature	Function-based index
C	Feature	Compressed index
R	Feature	Reverse key index
U	Uniqueness	Unique index that does not support a unique constraint.

For the most part only one of the indicators is needed. Rarely will more than one be used, perhaps in the case of a local unique index, or a global function-based index. See below for examples. There are other esoteric index features, such as hash-cluster indexes and domain indexes, but there is currently no planned or existing use of them in our systems, hence the limited list of type indicators.

NOTE: A global, non-partitioned index on a partitioned table should not use the L or G partitioning indicators. It is just a normal index that happens to be on a partitioned object.

Index Examples	**Explanation**
PHONE_PK PHONE_UK	Indexes that support the unique constraints on the PHONE table.
ORG_CNTRY_ID_IX	Simple index on a single, abbreviated column.
SU_COMP_IX, SU_COMP_IX2	Original and newly-added composite indexes on the SEC_USER table.
NL_COMP_CUIX	A compressed, unique composite index on the NATIONAL_LEADER table. (shows how index type indicators are concatenated)
FP_PERD_NM_FIX	A function-based index on the components of the period name in the FISCAL_PERIOD table.

LB_ACTIVE_YN_BIX	Bitmap index on the active flag in LOCAL_BANK.
CITIZEN_HOME_ST_ID_GIX	Globally-partitioned index on the home state in the CITIZEN table.
UD_COMP_UIX	Composite unique index on UNIT_DONATION (unique indexes should often be converted into unique key constraints.)

Oracle Context Index Objects

The author has not found much use for CTXCAT, CTXRULE or CTXXPATH indexes within his domain of expertise. If you do find a need for these, amend this doc internally and consider sending the improvements to the author maintaining the original in SourceForge.

However, Oracle CONTEXT indexes are immediately useful to most typical systems with text to search.

A basic Context index name is very similar to a regular index, excepting the suffix:

{Table Code_} {Column Descriptor} _CTX}

A Context index can have many modifiers specified in its create DDL, including wordlists, sectioners, lexers and so forth. There are so many options, in fact, that to attempt to reference them in the name would make the syntax unwieldy and the names unreadable. The only exception to this is datastores.

Use the Column Description component to indicate what sort of datastore is involved. The default DIRECT_DATASTORE uses the naming syntax above. But if the context index is a MULTI_COLUMN_DATASTORE create a name or abbreviation to represent the columns being indexed. For example, we frequently use a context index to achieve diacritic and case insensitive searches on first, middle and last name fields. So the token “fullname” is used as the column descriptor.

If the index is a FILE_DATASTORE, use the column descriptor component to indicate something identifiable about the directory or file being indexed.

If the index is a URL_DATASTORE, use the column descriptor to indicate something about the column that stores the internet addresses.

If the index is a DETAIL_DATASTORE, use the descriptor to indicate the table codes involved in the master-detail relationship.

Finally, if the index is a USER_DATASTORE, where a custom routine has been written to index single or multiple columns over multiple tables in a specialized manner, invent a name for the purpose of this index and use that as the column descriptor. For example, CC_ALL_TXT_FLD_CTX for a Context index that indexes all the text fields in the CUST_CONTACT table.

Sequence

A sequence name has the following components:

{Table Name} _SQ

Typically each table has a surrogate key that is fed by a single sequence generator dedicated to it. Therefore the sequence is named for that table.

Synonym

Synonyms follow no special convention. Synonyms mirror the name of the old item or remote item for which they serve as proxy. For example, if code depends on the STATE table at the remote MSTR database, the synonym will also be named STATE.

Type

A user-defined, schema-level type is named with the following components:

[System_] [Logical Group_] {Type Name} [_Collection Modifier] T

The Type Name is as the developer sees fit, but must conform to the abbreviation standard. The Collection Modifier[1] is "T" if the type is a collection (either of scalar datatypes or object types).

Examples would be STR_TT for a generic nested table of VARCHAR2 strings, CONTACT_T for a user-defined object type that contains multiple fields about user contact information, and CONTACT_TT (read Contact Type Table) for the corresponding collection type of the CONTACT_T type.

Queue

If using advanced queuing, these are the components that make up a queue name:

[System_] [Logical Group_] {Queue Name_} {Q | EQ}

The Queue Name is as the developer sees fit, within Oracle naming constraints and our abbreviation standard. The Queue Type is either "Q" for normal queues, or "EQ" for exception queues.

An example is IMAIL_Q, a normal queue based on IMAIL_SQT, a single-consumer queue table.

[1] We toyed with ARR, NTAB, NT, TAB and COLL as the type modifier. C was an option, standing for Collection type. For various reasons T was chosen, which is short for Table type and can represent both varrays and nested tables of scalar and user-defined types.

Stored PL/SQL

These are naming conventions for PL/SQL objects created by the database developers for their projects.

Package

The name components for PL/SQL packages are:

[System_] [Logical Group_] {Purpose} [_Function Code][_PKG]

The only required component of a package name is its purpose, a word that indicates the functional area of the routines it contains. Typical packages in medium to large systems will also use the optional system and logical grouping codes as well. Examples include: DUNS_FEED_ETL, AR_DLY_SWEEP, etc.

The optional Function Code is typically used when refactoring packages that are too long, doing too much. In this case, the original base name of the legacy package is kept, but each new modular package gets a short differentiating suffix. For example a large package named DB_MONITOR might be refactored into three, a DB_MONITOR_DML package for DML API routines, a DB_MONITOR_VLD for data validation routines, and a DB_MONITOR_NDS for native dynamic SQL cursors and routines.

The optional PKG object type modifier is a nod to legacy naming schemes. If package names are or will be quite long, forego the PKG suffix.

Packages that belong to the Starter database framework have very short names, partly because they belong to many systems, and partly because they are called so often, the short names reduce the amount of typing and indentation required. Examples include: CNST, TYP, DT, ENV, EXCP, IO, MAIL, MSGS, STR, NUM, LOGS and TIMER.

Procedure and Function

The two name components for packaged and standalone routines are verb and noun:

[Action_] {Purpose}

Since a function's purpose is usually to look up, determine, or return a value, the action will most often be "GET_", "FIND_" or "IS_". The action component for procedures is more diverse.

Here are some examples:

Functions: GET_LAST_LOG_TS, FIND_AVG_PRICE, IS_MEMBER, DEFAULT_LOG_PATH

Procedure: RUN_PROP_SYNC, ASSIGN_LEGAL, CREATE_STG_TABLES

NOTE:	Standalone procedures and functions are heavily discouraged. Consult countless PL/SQL books, gurus, online articles and the accompanying PL/SQL Programming Standard for an explanation as to why all PL/SQL should be in packages.

Trigger

Trigger naming conventions are as varied as there are triggers. These guidelines are the best we've seen after years of experience and trial & error.

DML Trigger

{Table Code_} [Purpose_] {B|A|CT [I|U|D] [R]|S} [#]

The designer has discretion over the code, column name or word used for the Purpose placeholder. It should be named such that it does not clash with other similar triggers on the same table.

The next component uses "B" for before triggers, "A" for after triggers and "CT" for 11g compound triggers. The "B" and "A" may be further coupled with "R" for row-level triggers and "S" for statement-level triggers. Row-level is the default if the "R" is missing from the name. If the trigger is the rare statement-level trigger, it must use "S" to identify itself as such.

Some shops like to identify the DML triggering events in the trigger name, using "I" for inserting, "U" for updating and "D" for deleting. We've found this tends to encourage lots of small triggers. This can be a problem since pre-11g Oracle doesn't guarantee the order of trigger firing. Instead, code one before row-level and one after-level, and keep all related processes in the same trigger so you can personally control the order.

The final component, the increment placeholder, is used when multiple triggers on the same table clash in name (usually those with WHEN clauses). As of 11g, this integer can also indicate trigger firing order.

Trigger naming examples could include:

SU_CHECK_MSTR_BR, SUPS_AS, SU_SYNC_TO_PFM_AR, POLP_SYNC_TO_EXT_A1, POLP_SYNC_TO_EXT_A2, SUGM_EMAIL_ADMIN_BS

Instead-Of Trigger

Naming an instead-of trigger is very straightforward, as "_TRG" is simply suffixed to the view name:

{View Name} _TRG

System Trigger

Naming a system trigger uses the following components:

{D|U} {_Triggering Event} _TRG

"D" stands for database-wide system triggers. "U" stands for user-level system triggers.

The Triggering Event is an immutable name as defined by the Oracle docs on system-level triggers.

Examples could include:

D_STARTUP_TRG, U_ALTER_TABLE_TRG, U_BEFORE_TRUNCATE_TRG, U_AFTER_DROP_TRG

Application Context

[System_] [Logical Group_] {Purpose} _CTX

Since the application context is actually owned by SYS, no matter what schema created it, it is a good idea to use the optional system and/or logical group codes to keep the application contexts from colliding in namespace with each other.

DBA-Centric Objects

The following items are usually not found in a data model, but are nevertheless items that require naming consistency.

Tablespace

A tablespace name uses the following components:

{System_} [Logical Group_] {Tablespace Type} [_##]

The Tablespace Type is typically either DATA or INDEX, but other types can be invented if the Database Engineer sees the need for special tablespaces, say for archival, maintenance or performance reasons.

Most tablespaces will simply be the system code or schema name (either serves as the Logical Group), followed by the type code. Examples for one system would be GM_DATA and GM_INDEX.

The optional increment number placeholder is used for very large partitioned objects where each partition requires its own tablespace. For example: PS_IMAGES_01, PS_IMAGES_02, etc.

Database Link

A database link is best named by using the system being linked to or its purpose. Do not use the database name or host name in the link name as these tend to change over time.

{Link Purpose} [_LINK]

Some examples would be NREL and IRS_LINK.

Odds and Ends

This document currently has no conventions or standards for stored Java, Roles, Libraries, Profiles, or Directories.