

Enterprise Data Standards

# Version Control

| **Version** | **Changes** | **Author** | **Date** |
| --- | --- | --- | --- |
| 0.1 | Initial draft | D Bennett | 18/02/2016 |
| 0.2 | Data Modelling Standards Included | D Bennett | 22/02/2016 |
| 0.3 | Conceptual Data Modelling Standards Included | D Bennett | 23/02/2016 |
| 0.4 | Further refinement | D Bennett | 02/03/2016 |
| 0.5 | Further refinement | D Bennett | 08/08/2016 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Table of Contents

[Purpose 4](#_Toc444694127)

[Scope 4](#_Toc444694128)

[Data Modelling 5](#_Toc444694129)

[Overview of Framework 5](#_Toc444694130)

[Enterprise Architecture Models & Objects 5](#_Toc444694131)

[Data Models 7](#_Toc444694132)

[Risk/Reward based approach 8](#_Toc444694133)

[Enterprise Architecture Data Modelling Standards 9](#_Toc444694134)

[Relational Modelling Standards 12](#_Toc444694135)

[External Data Modelling Standards 16](#_Toc444694136)

[Dimensional Modelling Standards 19](#_Toc444694137)

[Naming Standards 21](#_Toc444694138)

[Scope 21](#_Toc444694139)

[General Rules 21](#_Toc444694140)

[Entity Names 23](#_Toc444694141)

[Attribute Names 23](#_Toc444694142)

[Guidelines when constructing an attribute name 26](#_Toc444694143)

[Definitions & Comments 27](#_Toc444694144)

[Appendix 28](#_Toc444694145)

[Class Terms (Domains) 28](#_Toc444694146)

[Approved Acronyms, Initialisms & Abbreviations 29](#_Toc444694147)

[Normal Forms 30](#_Toc444694148)

[Definitions - ISO/IEC 11179-4:2004(E) 35](#_Toc444694149)

[Parts of Speech 40](#_Toc444694150)

[Reserved Words 41](#_Toc444694151)

[Glossary 57](#_Toc444694152)

# Purpose

This document provides guidelines and standards for data naming, data types and data modelling. It is intended to provide consistency across the disciplines, leading to greater compatibility and reuse of data, and fit with business reporting tools.

# Scope

The table below depicts the applicability of the rules and modelling standards to different types of information.

**Structured data** has a high degree of organisation, where its content is organised, and is machine readable. It includes relational database tables, sensor data, and most structured files, XML and JSON documents.

**Unstructured data** has less organisation, and often procedural processing is required to derive meaning from it using a computer. It includes documents, images, and blocks of text without defined context such as social media streams. Its structure cannot be modelled by traditional means.

| **Standard** | **Depicts** | **Type** | **Structured** | **Unstructured** |
| --- | --- | --- | --- | --- |
| Conceptual Data Model | Business objects (does not show structure) | Enterprise  Architecture | Yes | Yes |
| Application Entity Model | System objects (does not show structure) | Enterprise  Architecture | Yes | Yes |
| Enterprise Logical Data Model | Enterprise standard data structure | Data Architecture | Yes | No |
| Dimensional Model | Dimensional star schema | Data Architecture | Yes | No |
| External Data Modelling | Logical and Physical standards for procurement | Data Architecture | Yes | No |
| Application Logical Data Model | Application logical data structure | Data Architecture | Yes | No |
| Application Physical Data Model | Physical data structure | Data Architecture | Yes | No |
| Naming Standards | Standard naming conventions for entities and attributes | Rules | Yes | No |

# Data Modelling

## Overview of Framework



## Enterprise Architecture Models & Objects

### Enterprise Architecture Objects and Relationships

* **Business Layer: Business Object**
  + A system agnostic representation of something the business holds information about, such as customer, account, product
  + This is the level that will interoperate with:
    - Governance (data governance, records management, security, information privacy) in order to define policy and assign responsibilities
    - Business Analysis in order to define high-level business processes
* **Application Layer: Application Object**
  + The implementation of a business object in a specific system
  + In many cases each business object will exist in many systems
  + This is the level that will interoperate with:
    - Business Analysis in order to define lower level business processes
    - Solutions Architecture in order to model
      * Integration (mapping sources and targets)
      * Linkage to user interfaces, processes, capabilities and functions of applications
* **Technology Layer: Artefact**
  + The physical implementation of an application object on disk
  + In many cases each application object will be a grouping of files, or tables and columns in the application

### EA Relationships

1. Relationships between objects in different architecture layers is shown using a Realisation relationship.
   1. No name is necessary.
   2. Relationships should be via the immediate layer but can be inferred through two layers
   3. The direction of relationship is that the upper layer object (highest = business, application, technology = lowest) is realised by the lower layer object.
2. Relationships between objects in the same architecture layer should be shown using an Association relationship.
   1. A name is desirable.
   2. Association relationships have no direction.

### EA Data Management Objectives

When combined with the interoperable EA disciplines, management of data in this hierarchy enables:

* Impact Analysis for change, rationalisation and regulation
  + What data is where?
  + What processes does it drive?
  + What systems are dependent upon it?
  + What processes maintain and consume it?
  + Where does it go (integration)?
  + What technologies contain sensitive data?
  + Is there unnecessary duplication of data?
* Centralised decision making
  + Decisions made (legislation, policy) against the business object
  + Understanding where the policy change needs to be applied

### EA Data Standards

Enterprise Architecture standards are currently pending (tool and notation standards).

## Data Models

A data model is a visual depiction of the whole or part of the enterprise data landscape. It should have appropriate scope and content for its target stakeholders. Models will be produced according to the following standards:



The **Enterprise Architecture Models** (Conceptual, Application Entity) are the foundation of the Enterprise Taxonomy. This taxonomy is designed to maximise centralised decision making and ownership, to enable inheritance of decisions to lower levels and to provide insight and context to the data governance team.

**Conceptual Data Model(s)** depict the business objects within a subject area. It is a level at which interoperability will be managed against other Architecture functions (Business, Technology) and with Data Governance. Policy will be defined against business objects and inherited by the more granular logical and physical structures they are comprised of. Relationships are shown between business objects, containing verb-phrases. Sub/Super typing is permissible by nesting. Cardinality should not be shown. Many-to-many relationships are acceptable. The model does not show attributes. 1-100 business objects per subject area. Key metadata held against business objects includes a description, impact, risk, retention, disclosure, privacy & security policies, system of record, roles & responsibilities.

* The conceptual data model will comply with Conceptual Data Modelling Standards

**Application Entity Models** describe application objects. It is a level at which interoperability will be managed with other Architecture functions (Solutions, business). Relationships are defined to perform classification of application data objects to the business objects they represent, to link application data objects to interfaces and functions, and to show dependencies amongst application data. Sub/Super typing is permissible by nesting. Cardinality should not be shown. Many-to-many relationships are acceptable. The model does not show attributes. Key metadata held against application data objects includes a description.

* Application Entity Models will comply with Application Entity Modelling Standards

The **Enterprise Logical Data Model** is a system agnostic depiction of the structure of business data. It is the enterprise standard when building bespoke applications (including data warehousing) and the canonical message structure when building loose coupled integration. It should be comprehensive and cover all information for an entity/subject area. Relationships should be shown with cardinality and descriptive verb-phrases. Sub and super typing is shown. Attributes and keys are shown. Views can be produced by subject area, entity or for a specific need, but the model should be produced as a single view for the whole of the enterprise. Key metadata held against entities and attributes include names, descriptions, and the definition of good (values, foreign keys, rules and constraints), data types, length and precision.

* The enterprise logical data model will comply with relational modelling standards.

**Dimensional Models** depict a de-normalised view of data, optimised for reporting.

1. Dimensional models will comply with dimensional modelling standards.

**Application Logical and Physical Models** depict the structure of data within specific technologies. They will be produced where they add value and minimise risk. There is an expectation that vendors and systems integrators will provide and support this where appropriate, and our commercial agreements need to be updated to reflect this need. Where models are built on our behalf, they will use our preferred toolset and adhere to our standards. Where vendor supplied they will conform to a more generic set of minimum standards, which provide more flexibility in terms of notation and format. They should show structure of entities and tables, attributes, relationships including cardinality and recursive relationships, sub and super typing. Metadata as per the logical data model, with additional physical storage information.

* Where produced by or on behalf of PFG, the application logical and physical data models will comply with relational modelling standards.
* Where vendor provided, the application logical and physical data models will comply with minimum acceptable external modelling standards.

## Risk/Reward based approach

Where data models do not exist, use pragmatism to identify where they will add value. Ensure that data modelling is carried out where it reduces risk and maximises benefits, without unnecessarily incurring project costs and delays.

The following table identifies when data documentation (models and metadata) should be produced:

| **Nature of Solution** | **Definition** | **Documentation** |
| --- | --- | --- |
| Low impact | Standalone solution used by a small and non-critical area of the business. No reporting requirements. Not supported by PFG. | Not essential |
| Medium impact | Standalone solution used by a medium-critical area of the business, or integrated into a small ecosystem of non-critical systems. Possibly some reporting requirements. May be supported by PFG. | As required to implement and support |
| High impact | Solution impacting a critical business component. Possibly some reporting requirements. May be supported by PFG. | Mandatory |

## Enterprise Architecture Data Modelling Standards

### Conceptual Data Modelling Standards

A conceptual data model depicts business objects and their high level relationships.

1. A conceptual data model shows:
   * Business objects
   * Patterns (supertypes)
   * High level relationships
2. A business object is a system agnostic representation of something the business holds information about, such as customer, account, product
3. A business object can be structured or unstructured
4. Business objects need to be granular enough to interoperate with:
   * Governance (data governance, records management, security, information privacy) in order to define policy and assign responsibilities
   * Business Analysis in order to define high-level business processes
5. Business Objects should not take into account
   * Organisational divisions
   * Separation in source systems
6. Patterns (Supertypes) define common data formats and characteristics that can store multiple business objects in a standard manner, e.g. Customer, Colleague and Supplier are all forms of Party.
7. Patterns should not be too general and should be
   * Capable of being linked to standard processes
   * Capable of representation in a meaningful fixed structure logical data model with contained business objects differentiated by ‘type’, rather than ultra-flexible structures such as key-value pairs
8. Sub-Typing, Super-Typing and Pattern Inheritance is represented by enclosure within a parent object (nesting)
9. Relationships may contain a description, with an arrow indicating the direction in which it should be read depending on the modelling tool used.
10. Only key relationships need be shown
11. Borders may be used to show high-level groupings into subject areas
12. Conceptual data models should be produced according to brand or architecture guidelines
13. Business objects and patterns should be accompanied by metadata including:
    * Optional: The subject area in which the business object belongs
    * Mandatory: The name of the business object
    * Mandatory: The type of object (pattern or bus object)
    * If a business object conforms to a pattern, the name of the pattern
    * The description of the business object or pattern (compliant with definition standards)

### Application Entity Modelling

An application entity model depicts the data objects (entities) that are stored and used by an application. Depending on the enterprise architecture toolset, this can be built as a model, or generated from other architecture artefacts.

1. An application entity model shows:
   * Application data objects
   * Business objects
   * The realisation relationships between application data objects and business objects
   * Application object Patterns (supertypes)
   * High level relationships
   * Access relationships to external data
   * Interfaces
   * The relationship between interfaces and the business (in the case of external) or application data objects (in the case of internal) that is their payload
2. In the context of this document, the use of the term ‘application’ is synonymous with the Archimate definition of ‘application component’. A modular, deployable and replaceable part of a software system that encapsulates its behaviour and data and exposes these through a set of interfaces.
3. An application object is a system specific representation of something the application holds information about, such as customer, account, or product.
4. An application object may be comprised of structured and/or unstructured data.
5. Application data objects need to be granular enough to interoperate with:
   * Business processes describing user interface interactions
   * Applications architecture to define system processes
   * Integration
6. Application data objects should not show the underlying structure of the object it represents.
7. A good technique to identify application data objects is to look at application user manuals, user interfaces and interface specifications to identify the list of objects. The nouns in the name of the main dialog’s, and interfaces generally correspond to the application data objects.
8. Lookup tables and cross reference tables are generally not considered to be independent application data objects and should instead form part of the object they describe.
9. Patterns (Supertypes) define common data formats and characteristics that can store multiple application data objects in a standard manner, e.g. Customer, Colleague and Supplier are all forms of Party.
   * Patterns should not be too general and should be used where multiple application objects share standard application processes, data structures, interfaces
10. Sub-Typing, Super-Typing and Pattern Inheritance is represented by enclosure within a parent object (nesting), or a specialisation relationship where the modelling notation allows.
11. Relationships may contain a description, with an arrow indicating the direction in which it should be read and depending on the modelling tool used.
12. Only key relationships need be shown
13. Borders may be used to show high-level groupings into subject areas where the application is broad and complex
14. Application entity models should be produced according to brand and architecture guidelines
15. Application data objects should be named as follows (to make them unique):
    * Application - Object
    * E.g. “Pivotal – Customer”
16. Application interfaces should be named as follows (to make them unique):
    * Application – Object – Qualifier (a term to differentiate where multiple exist for the same object)
    * E.g. “Pivotal – Customer – Publish”
17. Application data objects and patterns should be accompanied by metadata including:
    * Mandatory: The name of the application data object
    * Mandatory: The type of object (pattern or business object)
    * If a business object conforms to a pattern, the name of the pattern
    * The description of the business object or pattern (compliant with definition standards)

Application data objects must be related to a business object via a Realisation relationship

## Relational Modelling Standards

### Scope

1. Relational Models apply only to structured data, in relational data sources (primarily databases).

### Notation

1. Relational data models should comply with Information Engineering (IE) notation standards, this is also known as ‘Crows Foot’ notation.

### Model Naming & Scope

1. Data models will be named according to the following standards:

ModelType + [Application +] [AppVersion +] [Qualifier +] [SubjectArea +] ModelVersion

ModelType denotes the type of data model:

CDM Conceptual Data Model

ELDM Enterprise Logical Data Model

ALDM Application Logical Data Model

PDM Physical Data Model

Application (Optional, CDM, ALDM and PDM only) is used to specify the system that is being modelled

AppVersion (Optional, CDM, ALDM and PDM only) is used to specify the version of the application that is being modelled.

Qualifier (Optional, ALDM and PDM only) is used to identify specific contexts within the application, for example differentiating between EDW staging and base layers or app modules.

SubjectArea denotes the scope of the model, this could be the name of an entity, project, business process or other context, or could specify ‘Full’ to denote that the model is comprehensive (for its ModelType, Application, Qualifier and AppVersion).

ModelVersion denotes the version of the model. This should be represented as a three part number, separated by periods.

a.b.c

aa Major release

bb Minor release

cc Working version

Non zero denotes work in progress versions

00 is the approved released version

Examples:

CDM Product Management 1.1.4

ELDM Full 1.1.0

ALDM Focus 10.2 Customer 12.1.0

ALDM EDW 10.1 Base Weekly Sales 1.1.0

### Entities

1. Entities are represented as a box, with the name of the entity in bold, and either (depending on the modelling tool used):
   1. Above and aligned to the top left corner (see example using CA ErWin)



* 1. In a ‘heading’ section at the top of the entity (example using Visio Crows foot notation)



### Primary Key Attributes

1. Attributes that make up the primary key should be entered (in key sequence), usually above a dividing line within the body of the entity
2. Displaying PK to denote primary key is optional

### General Attributes

1. Attributes that do not make up the primary key should be entered (in sequence) below a dividing line within the body of the entity
2. Attributes that are a foreign key (used to reference another entity) should be identified by the code (FK), e.g.



### Entity Metadata

1. All entities will be assigned a description
2. All entities will be related to their appropriate business object

### Relationships

1. Relationships should be shown according to Information Engineering notation
2. Relationships should (mandatory in logical models, optional in physical) contain phrases to describe the relationship in both directions.



1. Relationship Cardinality will be displayed on both ends of the relationship.
2. Relationship Cardinality will be displayed according to Information Engineering Notation

* Zero or One
  + The relationship maps to zero or one (but not more than one) records on the corresponding entity
  + Denoted as a Line and Circle



* One
  + The relationship maps to one (only one, and not zero) record on corresponding entity
  + Denoted as a Line (may be a double line in some modelling tools)



* One or Many
  + The relationship maps to one or many (not zero) records on the corresponding entity.
  + Denoted as a Line and Crows Foot (may be only a crows foot in some modelling tools)



* Zero or More
  + The relationship maps to zero, one or many records on the corresponding entity
  + Denoted as a Circle and Crows Foot (may also include a line depending on the modelling tool used)



1. Identifying & Non-Identifying Relationships will be shown according to Information Engineering Notation:
   * Identifying Relationship
     + Where one or more of the primary key fields of a dependant entity is a primary key of another entity (i.e. the entity cannot be defined alone).
     + Represented as a Solid Line



* + Non-Identifying Relationship
    - Where the all components of the primary key represent only the entity that is being defined (i.e. the entity can be defined alone)
    - Represented as a Dashed Line

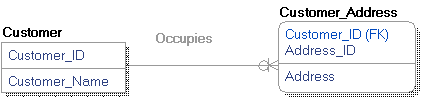


1. Relationship Cardinality is direction sensitive. The symbol is entered at the end of the line adjacent to the entity to which the cardinality applies:

* Example



* Reading Left to Right



A customer occupies *Zero, One or More* Customer\_Address(es)

* Reading Right to Left



A Customer\_Address is occupied by *One* Customer

### Normalisation

Normalisation is the process of removing data redundancy and structuring data in a manner that is optimised for relational processing. See appendix for definition of Normal Forms.

1. Logical models should be normalised to the Third Normal Form, or greater.
2. Physical models may be de-normalised for performance reasons
3. Dimensional models should be de-normalised to no less than Second Normal Form for usability and performance reasons

## External Data Modelling Standards

As suppliers do not have access to PFG’s modelling standards and toolset, and there is no benefit in re-engineering third party products to comply with the standards, the following set of minimum acceptable modelling standards apply, and will be used in procurement evaluations.

The following standards apply where data is held in a relational format.

### Standards

1. The following data modelling notations are accepted:
   * Information Engineering (IE)
   * Integration Definition for Information Modelling (IDEF1X)
   * Barker’s Notation
   * Unified Modelling Language (UML)
2. Other industry standard notations may be accepted based on consultation with data architecture
3. Models will comply with the standards of the notation in which they are presented
4. Models will be appropriately supported
5. Models will be structured and organised
   * Neatly, minimising overlapping lines
   * In appropriate groupings (subject areas) if large
6. Models will show:
   * Entities
   * Attributes
   * Primary key(s)
   * Relationships between entities, including
     + Cardinality of relationships (how many to how many)
     + Whether the relationship is mandatory or optional
     + Foreign key(s)
7. Models will be accompanied by metadata describing
   * Each Entity
     + Its name
     + Its description
     + An indication of the functionality driven by the entity
     + Whether the entity is a table or view
   * Each Attribute
     + Its name
     + Its description
     + Whether the attribute is mandatory or optional
     + Its datatype, length, precision and scale
     + Any constraints that apply to the attribute
     + An indication of the functionality driven by the entity

### Required Documentation

1. It is recognised that some vendors will not have documentation available to appropriate standards. In the event that it is not available, the effort of producing the documentation should be considered to be part of the total cost of ownership for the product. Production of missing documentation will take place based on risk and reward:

| **Nature of Solution** | **Definition** | **Documentation** |
| --- | --- | --- |
| Low impact | Standalone solution used by a small and non-critical area of the business. No reporting requirements. Not supported by PFG. | Not essential |
| Medium impact | Standalone solution used by a medium-critical area of the business, or integrated into a small ecosystem of non-critical systems. Possibly some reporting requirements. May be supported by PFG. | As required to implement and support |
| High impact | Solution impacting a critical business component. Possibly some reporting requirements. May be supported by PFG. | Mandatory |

1. Documentation required from a vendor will be appropriate to the type of product provided:

| **Type of Product** | **Definition** | **Models & Metadata** | **User**  **Manuals** | **API**  **Manuals** |
| --- | --- | --- | --- | --- |
| Outsourced Business Service (with Systems Access and/or integration) | A standalone "Black Box" Business service that performs a function on behalf of the company, with the partner having access to some of CCD’s existing systems (eg:- e-mail) or some information touch points  Eg:- Security at Head Office | N/A | N/A | N/A |
| Tool or Infrastructure | A piece of software, technology or a service that either  (a) performs generic infrastructure or software services, or  (b) an IT technology that enables systems development  -Visible to specialist company users  Eg:- SQL Server | Required  (for config/ metadata) | Required  (Technical) | Required |
| Data Service | A "Black Box" Business service that is:  - Not visible to company users  - Wholly controlled by its vendor  - Wholly hosted by the same vendor  - Wholly supported by the same vendor  - Has integration and/or data migration touch-points with the company  Eg:- QAS Address Validation | N/A | N/A | Required |
| Software As A Service (SAAS) | "A ""Black Box"" IT solution that is:  - Visible to company users  - Wholly controlled by its vendor  - Wholly hosted by the same vendor  - Wholly supported by the same vendor  Eg:- RemedyForce | N/A | Required | Required |
| Packaged Solution with internal or external support | A COTS IT solution that is not hosted by its provider, it is:  - Visible to company users  - Wholly controlled by its vendor  - Wholly supported by the vendor, a partner or PFCCD  Eg:- CODA Financials | Required | Required | Required |
| Development Accelerator | A MOTS IT solution that is intended to be modified for use.  Eg:- Microsoft Dynamics | Required | Required | Required |
| Bespoke Solution(or Mod or Extension of Usage) | An IT solution or change to an existing IT solution that is not commercially available and is created by or on behalf of The company.  Eg:- Focus | Required | Required | Required |

## Dimensional Modelling Standards

Dimensional models are used to represent reporting data in a manner that is optimised for reporting. Dimensional models can follow either the snowflake or star-schema pattern.

### General Rules

1. Dimensional models may be substituted for BEAM\* tables if this method is used to gather the requirements
2. The name of a dimensional model (the subject area) *should* correspond to the name of the central fact table
3. Dimensional models *should* be arranged as a subject area with a single central ‘fact’ table, surrounded by dimensions.
4. Cardinality of relationships is represented as a directional arrow, with the ‘line’ end representing ‘one’, and the ‘arrow’ representing a (zero-)one-or-many relationship.
5. Optional (zero) relationships should be avoided. Replace null values with rows in dimension tables that have meaningful values on users’ reports, and that avoid null handling challenges in relational databases
   * Use negative surrogate keys to represent these values (e.g. -1 = “Unknown”)
6. Identify specific properties of attributes with the following annotations:
   * Primary key (PK)
   * Foreign key (FK)
7. If the modelling tool allows, identify specific properties of attributes with the following annotations:
   * Natural key (NK), the business key of an entity, which should be replaced by a surrogate
   * Degenerate Dimension (DD), the business key of a de-normalised parent-child fact.

### Star Schema (Dimensional) models

1. A star schema is more performant and user friendly, but results in greater data redundancy than a snowflake schema.
2. In a star schema, all dimensions must relate directly to the central fact, either by:
   1. Attaching ID’s of outlying dimensions to the central fact, or
   2. Collapsing attributes of the outlying facts into the immediate facts



1. Star Schemas should comply with Kimball’s ‘ten commandments’:
2. Load detailed atomic data into dimensional structures.

*Store data at its lowest level of granularity to enable it to be rolled up according to business needs.*

1. Structure dimensional models around business processes.

*Represent business processes as a central fact table within a star schema.*

1. Ensure that every fact table has an associated date dimension table.

*Every fact table should relate to a date dimension.*

1. Ensure that all facts in a single fact table are at the same grain or level of detail.

*Ensure all sources are capable of providing data at the same level of granularity.*

1. Resolve many-to-many relationships in fact tables.

*Create dual-keyed bridge tables between facts and dimensions if necessary.*

1. Resolve many-to-one relationships in dimension tables.

*De-normalise dimension tables for performance.*

1. Store report labels and filter domain values in dimension tables.

*Assign codes to substitute nulls. Ensure these are mapped into dimension tables.*

1. Make certain that dimension tables use a surrogate key.

*Assign surrogate keys to all entities. Ensure they are sequential and meaningless.*

1. Create conformed dimensions to integrate data across the enterprise.

*Standardise dimensions across all facts.*

1. Continuously balance requirements and realities to deliver a DW/BI solution that’s accepted by business users and that supports their decision-making.

*Delivery of a working solution is preferred to the non-delivery of a perfect solution.*

### Snowflake Schema

1. A snowflake schema is less performant and less user-friendly than a star schema, but is more normalised with less data redundancy.
2. In a snowflake schema, dimensions can relate to a fact via intermediate dimensions.
3. Where possible, star schemas are preferred to snowflakes.



# Naming Standards

## Scope

The rules in this document apply to:

1. Logical Data Models except:
   1. Where provided by the vendor of a COTS (Commercial Off The Shelf) application
   2. Where the model represents data sourced from a COTS solution (such as the landing area of a data warehouse)
2. Physical Data Models and resultant schemas except:
   1. Where provided by the vendor of a COTS (Commercial Off The Shelf, not developed specifically by or for the company) application
   2. Where the model represents data sourced from a COTS solution (such as the landing area of a data warehouse)
3. SOA (Service Oriented Architecture) business messages and API’s except:
   1. ABM’s (Application Business Messages) that integrate to or from a COTS solution.

## General Rules

### Acronyms & Abbreviations

Acronyms and abbreviations can lead to confusion if not properly managed. A register of approved terms will be produced, non-approved terms may not be used.

1. Unapproved acronyms *must not* be used (see appendix).
2. Unapproved abbreviations *must not* be used (see appendix).

### Naming and Semantics

1. Names *must* be
   1. Meaningful, allowing a user to identify what the element represents
   2. Stated in the singular (except where a specific class term permits)
   3. Concise
   4. Verbs should be stated in the present tense
   5. Able to stand alone
   6. State what the concept is, rather than what it is not
2. Intersect (Junction) entities are used to avoid many to many relationships.
   1. *Where possible*, avoid the names of the intersected entities if a business term exists to describe the intersection

### Characters & Case

1. The first character of any name *must* be an upper case letter (A-Z)
2. Names *must* only contain the following characters:
   1. Lower case letters (a-z)
   2. Upper case letters (A-Z)
   3. Numbers (0-9)
   4. Underscores
3. Terms *should* be separated:
   1. Where technology permits, underscores “\_” *must* be used as the separator.
   2. Where technology does not permit underscores:
      1. Alternative characters *may* be substituted (separators *must* be single characters, consistent across the technology, and must not be letters or numbers).
      2. Underscores *may* be removed (relying on camel or pascal case to identify separate terms).
4. Logical Names *must*, and Physical Names *should* (technology permitting) adopt the following case rules:
   1. Non-Acronym/Initialism terms in camel or pascal case
   2. Acronym/Initialism terms in upper case (all capitals)
5. Names (regardless of logical or physical) *should* be capable of remaining unique when:
   1. Converted to all capitals
   2. Separators are removed.

### Content

1. Names *must not* contain (see the ‘parts of speech’ appendix for definitions):
   1. Articles (A, An, The)
   2. Interjections (Hey, Wow)
   3. Conjunctions (And, Or)
   4. Pronouns (She, We, They, This, That)
   5. Prepositions (About, At, Down, Of, With)
   6. Quantifiers (Much, Many, Lots)
2. Names *should not* contain:
   1. Adverbs (Gently, Almost, Helpfully)

### Length

1. Physical attribute lengths *must not* exceed 63 characters (to enable portability/replication to MySQL PostgreSQL databases)
2. Physical attribute lengths *should not* exceed 30 characters (to enable portability/replication of data to Oracle databases)
3. Shorter name lengths reduce the additional element metadata sent in XML and JSON messages, display better on report headings and require less processing, storage and network bandwidth. Avoid unnecessary length *where possible*.
4. In the event that lengths cannot be reduced, request additional abbreviations or acronyms from the data architecture function.

### Reserved Words

1. Names (in their entirety) *must* *not* conflict with reserved words in (see appendix):
   1. Major Standards
      1. ODBC
      2. ANSI SQL
   2. Major Data Technologies
      1. SQL Server
      2. MySQL and derivatives
      3. Oracle
      4. DB2
      5. PostgreSQL and derivatives

## Entity Names

### Structure of an Entity Name

Entity names will be structured in the following sequence:

1. Prime Business Term(s)
   1. Mandatory
   2. Noun(s) that represent the entity
   3. Named in the singular
   4. With the most significant term first
2. Qualifier or Modifier Terms
   1. Optional, used where there is a need to differentiate between or clarify multiple prime term/class term combinations.
   2. Usually Adjectives or Nouns
   3. Used where the class term alone is not sufficient to differentiate
   4. Named in the singular

### Data Warehouse Specific Rules

1. Data Warehouse Layer Information (Configuration, Landing, Staging, Base, etc) *should not* be embedded in table names (use a schema/user/database to perform separation)
2. In the presentation layer:
   1. Fact tables will be prefixed with Fact\_
   2. Dimension tables will be prefixed with Dim\_
3. In data warehouse fact tables, plural names are acceptable where the entity is an aggregate fact

### General Rules

1. Entity names should not include unnecessary terms such as:
   1. Data
   2. Information
   3. Details

## Attribute Names

### Structure of an Attribute Name

Attribute names will be structured in the following sequence:

1. Prime Business Term(s)
   1. Mandatory
   2. Noun(s) that represents the entity to which the attribute relates
   3. Named in the singular

*Embedding the entity name in each attribute enables business users to perform natural joins in analytics tools, and reduces errors caused by ambiguity*

1. Qualifier or Modifier Terms
   1. Optional, used where there is a need to differentiate between or clarify multiple prime term/class term combinations.
   2. Usually Adjectives or Nouns
   3. Used where the class term alone is not sufficient to differentiate
   4. Named in the singular
2. Class Term (or Domain) – A word that indicates the data type, usually a Noun.
   1. Mandatory
   2. This *must* (with the exception of date and time related data types) be a business term rather than a base datatype)
   3. This *should* describe an enterprise standard data format corresponding to the approved list of class terms (below)
   4. This *must* be named in the singular except where an approved class term exists that is designed to hold multiple values (e.g. Forenames)
3. Where the name uses Industry and generally Recognised Terms that do not comply with the above standards
   1. Treat the whole of the industry or recognised term as a qualifier or modifier term

### Foreign Keys

Foreign keys are fields used to join to another table.

1. Where the attribute name remains unique, adopt the full name of the attribute that is the origin of the foreign key attribute (this will generally be the entire primary key of a base entity)



*In the example above the foreign key field name is identical in both tables*

*Aligning the names of attributes used to perform joins enables business users to perform natural joins in analytic tools*

1. Where the adoption of the full attribute name creates a duplicate (the most common instances this would occur in are recursive relationships to the same table, or multiple relationships from the same table), include a qualifier or modifier term



### Audit Fields

1. Audit fields such as update dates and times will comply with attribute naming rules in full.
2. It is especially important to include the prime business term as the start of the audit field name (e.g. *Customer\_*Last\_Update\_Date, rather than Last\_Update\_Date)

*Differentiating the names of attributes not intended to perform joins reduces the risk of user error when performing natural joins in analytic tools*



### One Attribute: One Meaning

1. Attributes (except where in extensible data structures such as key-value pairs identified by a primary key that defines its context) that can be assigned multiple descriptions without identifying context should be avoided, they should be separated

*E.g. a general identifiers field (left) that could hold national insurance, driving licence number, etc is not appropriate as no context is provided without interrogating the data. A more appropriate structure is shown on the right, a system can use identifier\_type to get the required type of identifier without interrogating the attribute contents.*



1. Attributes that consist of multiple segments should be avoided unless there is absolute certainty that the separate components will never be used alone.

*E.g. A code that is made up of Brand and Product identifiers should be avoided:*

*Product: SAT-1234*

*Instead, break into two attributes:*

*Brand: SAT*

*Product: 1234*

### Domain Specific Rules

#### Boolean (Yes/No)

1. Where an attribute is stored in a Boolean (Yes/No) field, the following rules apply:
   1. Naming must be in the affirmative

*Customer\_Deceased\_Ind, not Customer\_Not\_Deceased\_Ind*

* 1. Naming must take into account null values (assumed to be no by default)

*Customer\_Deceased\_Ind, not Customer\_Alive\_Ind*

* 1. Naming should describe a state rather than an event

*Customer\_Deceased\_Ind, not Customer\_Died\_Ind*

## Guidelines when constructing an attribute name

1. Describe the element without acronyms, abbreviations or initialisms

***The surname of the manager of the project***

***The date on which the agreement begins***

***The minimum annual percentage rate of the loan***

1. Adjust the tense of the statement so that verbs, adjectives and adverbs fall between nouns where possible

***The surname of the manager of the project*** *(no change)*

***The date that is the beginning of the agreement*** *(beginning placed before agreement)*

***The annual percentage rate that is the minimum for the product*** *(minimum moved)*

1. Eliminate Articles, Interjections, Conjunctions, Pronouns, Prepositions, Quantifiers and Adverbs where there is no need to differentiate terms

*~~The~~* ***Name*** *~~of the~~* ***Manager*** *~~of the~~* ***Project***

*~~The~~* **date** *~~that is the~~* **beginning** *~~of the project~~* **agreement**

*~~The~~***annualpercentage rate***~~that is the~~* ***minimum*** *~~for the~~***product**

1. Reverse the sequence

**project manager name**

**project beginning date**

**product minimum annual percentage rate**

1. Identify if there are any approved abbreviations, acronyms or initialisms, and see if any substitute/common terms can be applied to longer/unusual words for consistency. Validate class terms against the approved list, and assess if a new one is needed for any gaps.

**project manager name** (no change)

**agreement start** ~~beginning~~ **date** (start substituted for beginning)

**product min APR** ~~annual percentage rate~~ (APR substituted for Annual Percentage Rate)

1. Apply case rules and separators

**Project\_Manager\_Name**

**Agreement\_Start\_Date**

**Product\_Min\_APR**

# Definitions & Comments

1. Definitions *must* be provided for every element.
2. Definitions *must* comply with ISO/IEC 11179-4:2004(E) – see appendix
3. Definitions *should* (technology permitting) accompany data, implemented as table and column comments and in business contracts.

# Appendix

## Class Terms (Domains)

| **Domain** | **Description** | **Data Type** | **Rules & Comments** |
| --- | --- | --- | --- |
| ID | Identifier, a field assigned as a surrogate key, or a foreign key reference to non-lookup data | Integer | Signed. Unique in reference table, can have many values |
| Code | Reference to a lookup table | Small Integer | Signed. Unique in reference table, should not have many values |
| Ind | Indicator, a yes/no flag | Boolean |  |
| Name | Name field | Varchar (64) |  |
| Comments,  Details | General comments field | Varchar (2048) |  |
| Descr | A description field | Varchar (256) |  |
| DOB | Date of Birth | Date | Cannot be future |
| Date | A date | Date |  |
| Time | A time | Time |  |
| Datetime | A date and time | DateTime |  |
| Days | A number of days | Smallint |  |
| Balance,  Amount | A financial value | Decimal (12,4) | Signed |
| Rate | Percentage | Decimal (8,4) | Signed |
| Percent | Percentage | Decimal (8,4) | Signed |
| URL,  URI,  URN | A web address (by name, by location or by identifier) | Varchar(2083) |  |
| Value | A generic value field that can hold any datatype | Varchar(2048) |  |
| Token | A code that identifies a permission | Varchar(2048) |  |

Where datatypes need to serve web applications, the following substitutions may be made in terms of datatypes:

* Varchar may be substituted for nvarchar if it is expected that the whole range of UTF-8 (Unicode) characters need to be stored.

## Approved Acronyms, Initialisms & Abbreviations

| **Acronym** | **Replaces** | **Reason** |
| --- | --- | --- |
| APR | Annual Percentage Rate | Extremely Common |
| DOB | Date Of Birth | Extremely Common |
| ID | Identifier | Extremely Common |
| Amt | Amount | Extremely Common |
| Avg | Average | Extremely Common |
| Descr | Description | Extremely Common |
| Ind | Indicator | Extremely Common |
| Max | Maximum | Extremely Common |
| Min | Minimum | Extremely Common |
| Qty | Quantity | Extremely Common |
| Ref | Reference | Extremely Common |
| Wk | Week | Extremely Common |
| Wkly | Weekly | Extremely Common |
| Mth | Month | Extremely Common |
| Mthly | Monthly | Extremely Common |
| Qtr | Quarter | Extremely Common |
| Qtrly | Quarterly | Extremely Common |
| Yr | Year | Extremely Common |
| Yrly | Yearly | Extremely Common |
| Dim | Dimension | Extremely Common |
| Pct | Percentage | Extremely Common |
| XRef | Cross Reference | Verbose |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Normal Forms

Normalisation is the process of rationalising data to minimise data redundancy, and maximise its effectiveness with the relational engine behind most databases. This appendix presents a (simplified) view of the most common normal forms.

### First Normal Form (1NF)

#### Simple Definition:

“The Key”

The entity has a primary key

The non-key attributes depend on the primary key

Each record should contain the same number of fields

There should be no repeating fields

#### Full Definition:

A relation is in first normal form if and only if the domain of each attribute contains only atomic (indivisible) values, and the value of each attribute contains only a single value from that domain.

#### Not 1NF:



*The record has two Car\_Reg fields*

#### 1NF:



*Split into separate entity*

### Second Normal Form (2NF)

#### Simple Definition:

“The Whole Key”

First Normal Form +

If the primary key is composite (made of multiple attributes), each non key attribute must depend on the whole of the key

#### Full Definition:

A table is in 2NF if it is in 1NF and no non-prime attribute is dependent on any proper subset of any candidate key of the table. A non-prime attribute of a table is an attribute that is not a part of any candidate key of the table.

#### Not 2NF:



*Fuel type does not relate to customer ID*

#### 2NF:



*Split into separate entity*

### Third Normal Form (3NF)

#### Simple Definition:

“Nothing But The Key”

Second Normal Form +

The non-key attributes must not depend on any non-primary-key attribute

#### Full Definition:

A table is in 3NF if and only if both of the following conditions hold:

* The relation R (table) is in second normal form (2NF)
* Every non-prime attribute of R is non-transitively dependent on every key of R.

#### Not 3NF:



Fuel Type is dependant upon model and make

#### 3NF:



*Split into separate entity*

### Boyce-Codd Normal Form (BCNF ,3.5NF)

#### Simple Definition:

“Nothing But The Key and All Candidate Keys”

Third Normal Form +

The non-key attributes must not depend on any non-prime candidate key

#### Full Definition

A relational schema R is in Boyce–Codd normal form if and only if for every one of its dependencies X → Y, at least one of the following conditions hold:

* X → Y is a trivial functional dependency (Y ⊆ X)
* X is a super key for schema R

#### Not BCNF:



*Engine ID is an alternate candidate key, and engine type is dependant upon it*

#### BCNF:



*Split into separate entity*

### Fourth Normal Form (4NF)

#### Simple Definition:

“No independent multi-value facts”

Third (or BCNF) Normal Form +

A record should not contain two independent multi-value facts

#### Full Definition:

A Table is in 4NF if and only if, for every one of its non-trivial multivalued dependencies X ->> Y, X is a superkey, that is, X is either a candidate key or a superset thereof.

#### Not 4NF:



*Owner\_Name and Tyre\_Type can both have multiple values, but are unrelated*

#### 4NF:



*Split into separate entities*

### Fifth Normal Form (5NF)

#### Simple Definition:

“The Entity consists of key and non key attributes only”

Fourth Normal Form +

The entity cannot be reconstructed from several smaller record types.

Fifth normal form is similar to fourth normal form and in many cases yields the same result, but in situations where symmetric constraints exist, it requires the creation of a bridge table to enforce the construct.

#### Full Definition:

A table is said to be in the 5NF if and only if every non-trivial join dependency in it is implied by the candidate keys.

#### Not 5NF:

4NF representation



*Assuming that certain tyres are only available with specific wheels*

*The correlation (symmetric constraint) between wheel\_type and tyre\_type cannot be established.*

#### 5NF:



*Additional entity added to enforce the relationship between wheel and tyre*

### Sixth Normal Form (6NF)

#### Simple Definition:

“Decomposed to Irreducible Components”

Fifth Normal Form +

Satisfies no non-trivial join dependencies at all.

Sixth normal form is generally used when temporal influences allow attributes to change at different rates and there is a desire to track each independently.

#### Full Definition:

A relvar R [table] is in sixth normal form (abbreviated 6NF) if and only if it satisfies no nontrivial join dependencies at all — where, as before, a join dependency is trivial if and only if at least one of the projections (possibly U\_projections) involved is taken over the set of all attributes of the relvar [table] concerned.

#### Not 6NF:



The Change\_Date column applies to both Job\_Title and Name

#### 6NF:



*Split into separate entities*

## Definitions - ISO/IEC 11179-4:2004(E)

### Summary

**A data definition shall:**

* Be stated in the singular
* State what the concept is, not only what it is not
* Be stated as a descriptive phrase or sentence(s)
* Contain only commonly understood abbreviations
* Be expressed without embedding definitions of other data or underlying concepts

**A data definition should:**

* State the essential meaning of the concept
* Be precise and unambiguous
* Be concise
* Be able to stand alone
* Be expressed without embedding rationale, functional usage, or procedural information
* Avoid circular reasoning
* Use the same terminology and consistent logical structure for related definitions
* Be appropriate for the type of metadata item being defined

### Detail

#### Be stated in the singular

Explanation: The concept expressed by the data definition shall be expressed in the singular. (An exception is made if the concept itself is plural.)

Example: “Article Number”

Good definition: A reference number that identifies an article.

Poor definition: Reference number identifying articles.

Reason: The poor definition uses the plural word “articles,” which is ambiguous, since it could imply that an “article number” refers to more than one article.

#### State what the concept is, not only what it is not

Explanation: When constructing definitions, the concept cannot be defined exclusively by stating what the concept is not.

Example: “Freight Cost Amount”

Good definition: Cost amount incurred by a shipper in moving goods from one place to another.

Poor definition: Costs which are not related to packing, documentation, loading, unloading, and insurance.

Reason: The poor definition does not specify what is included in the meaning of the data.

#### Be stated as a descriptive phrase or sentence(s)

Explanation: A phrase is necessary (in most languages) to form a precise definition that includes the essential characteristics of the concept. Simply stating one or more synonym(s) is insufficient. Simply restating the words of the name in a different order is insufficient. If more than a descriptive phrase is needed, use complete, grammatically correct sentences.

Example: “Agent Name”

Good definition: Name of party authorized to act on behalf of another party.

Poor definition: Representative.

Reason: “Representative” is a near-synonym of the data element name, which is not adequate for a definition.

#### Contain only commonly understood abbreviations

Explanation: Understanding the meaning of an abbreviation, including acronyms and initialisms, is usually confined to a certain environment. In other environments the same abbreviation can cause misinterpretation or confusion. Therefore, to avoid ambiguity, full words, not abbreviations, shall be used in the definition.

Exceptions to this requirement may be made if an abbreviation is commonly understood such as “i.e.” and “e.g.” or if an abbreviation is more readily understood than the full form of a complex term and has been adopted as a term in its own right such as “radar” standing for “radio detecting and ranging.”

All acronyms must be expanded on the first occurrence.

Example: “Tide Height”

Good definition: The vertical distance from mean sea level (MSL) to a specific tide level.

Poor definition: The vertical distance from MSL to a specific tide level.

Reason: The poor definition is unclear because the acronym, MSL, is not commonly understood and some users may need to refer to other sources to determine what it represents. Without the full word, finding the term in a glossary may be difficult or impossible.

Example2 - “Unit of Density Measurement”

Good definition: The unit employed in measuring the concentration of matter in terms of mass per unit (m.p.u.) volume (e.g., pound per cubic foot; kilogram per cubic meter).

Poor definition: The unit employed in measuring the concentration of matter in terms of m.p.u. volume (e.g., pound per cubic foot; kilogram per cubic meter).

Reason: m.p.u. is not a common abbreviation, and its meaning may not be understood by some users.

The abbreviation should be expanded to full words.

#### Be expressed without embedding definitions of other data or underlying concepts

Explanation: As shown in the following example, the definition of a second data element or related concept should not appear in the definition proper of the primary data element. Definitions of terms should be provided in an associated glossary. If the second definition is necessary, it may be attached by a note at the end of the primary definition's main text or as a separate entry in the dictionary. Related definitions can be accessed through relational attributes (e.g., cross-reference).

Example 1: “Sample Type Code”

Good definition: A code identifying the kind of sample.

Poor definition: A code identifying the kind of sample collected. A sample is a small specimen taken for testing. It can be either an actual sample for testing, or a quality control surrogate sample. A quality control sample is a surrogate sample taken to verify results of actual samples.

Reason: The poor definition contains two extraneous definitions embedded in it. They are definitions of “sample” and of “quality control sample.”

Example 2: "Issuing Bank Documentary Credit Number"

Good definition: Reference number assigned by issuing bank to a documentary credit.

Poor definition: Reference number assigned by issuing bank to a documentary credit. A documentary credit is a document in which a bank states that it has issued a documentary credit under which the beneficiary is to obtain payment, acceptance, or negotiation on compliance with certain terms and conditions and against presentation of stipulated documents and such drafts as may be specified.

Reason: The poor definition contains a concept definition, which should be included in a glossary.

#### State the essential meaning of the concept

Explanation: All primary characteristics of the concept represented should appear in the definition at the relevant level of specificity for the context. The inclusion of non-essential characteristics should be avoided. The level of detail necessary is dependent upon the needs of the system user and environment.

Example 1: “Consignment Loading Sequence Number” (Intended context: any form of transportation)

Good definition: A number indicating the sequence in which consignments are loaded in a means of transport or piece of transport equipment.

Poor definition: A number indicating the sequence in which consignments are loaded in a truck.

Reason: In the intended context, consignments can be transported by various transportation modes,

e.g., trucks, vessels or freight trains. Consignments are not limited to trucks for transport.

Example 2: “Invoice Amount”

Good definition: Total sum charged on an invoice.

Poor definition: The total sum of all chargeable items mentioned on an invoice, taking into account deductions on one hand, such as allowances and discounts, and additions on the other hand, such as charges for insurance, transport, handling, etc.

Reason: The poor definition includes extraneous material.

#### Be precise and unambiguous

Explanation: The exact meaning and interpretation of the defined concept should be apparent from the definition. A definition should be clear enough to allow only one possible interpretation.

Example: “Shipment Receipt Date”

Good definition: Date on which a shipment is received by the receiving party.

Poor definition: Date on which a specific shipment is delivered.

Reason: The poor definition does not specify what determines a “delivery.” “Delivery” could be understood as either the act of unloading a product at the intended destination or the point at which the intended customer actually obtains the product. It is possible that the intended customer never receives the product that has been unloaded at his site or the customer may receive the product days after it was unloaded at the site.

#### Be concise

Explanation: The definition should be brief and comprehensive. Extraneous qualifying phrases such as “for the purpose of this metadata registry,” “terms to be described,” shall be avoided.

Example: “Character Set Name”

Good definition: The name given to the set of phonetic or ideographic symbols in which data is encoded.

Poor definition: The name given to the set of phonetic or ideographic symbols in which data is encoded, for the purpose of this metadata registry, or, as used elsewhere, the capability of systems hardware and software to process data encoded in one or more scripts.

Reason: In the poor definition, all the phrases after “...data is encoded” are extraneous qualifying phrases.

#### Be able to stand alone

Explanation: The meaning of the concept should be apparent from the definition. Additional explanations or references should not be necessary for understanding the meaning of the definition.

Example: “School Location City Name”

Good definition: Name of the city where a school is situated.

Poor definition: See “school site”.

Reason: The poor definition does not stand alone, it requires the aid of a second definition (school site) to understand the meaning of the first.

#### Be expressed without embedding rationale, functional usage, domain information, or procedural information

Explanation: Although they are often necessary, such statements do not belong in the definition proper because they contain information extraneous to the definition. If deemed useful, such expressions may be placed in other metadata attributes (see ISO/IEC 11179-3). It is, however, permissible to add examples after the definition.

1) The rationale for a given definition should not be included as part of the definition (e.g. if a data element uses miles instead of kilometres, the reason should not be indicated in the definition).

2) Functional usage such as: “this data element should not be used for ...” should not be included in the definition proper.

3) Remarks about procedural aspects. For example, “This data element is used in conjunction with data element 'xxx'”, should not appear in the definition; instead use “Related data reference” and

“Type of relationship” as specified in ISO/IEC 11179-3.

Example: “Data Field Label”

Good definition: Identification of a field in an index, thesaurus, query, database, etc.

Poor definition: Identification of a field in an index, thesaurus, query, database, etc., which is provided for units of information such as abstracts, columns within tables.

Reason: The poor definition contains remarks about functional usage. This information starting with “which is provided for...” must be excluded from the definition and placed in another attribute, if it is necessary information.

#### Avoid circular reasoning

Explanation: Two definitions shall not be defined in terms of each other. A definition should not use another concept's definition as its definition. This results in a situation where a concept is defined with the aid of another concept that is, in turn, defined with the aid of the given concept.

Example: two data elements with poor definitions:

1) Employee ID Number - Number assigned to an employee.

2) Employee - Person corresponding to the employee ID number.

Reason: Each definition refers to the other for its meaning. The meaning is not given in either definition.

#### Use the same terminology and consistent logical structure for related definitions

Explanation: A common terminology and syntax should be used for similar or associated definitions.

Example: The following example illustrates this idea. Both definitions pertain to related concepts and therefore have the same logical structure and similar terminology.

1) “Goods Dispatch Date” - Date on which goods were dispatched by a given party.

2) “Goods Receipt Date” - Date on which goods were received by a given party.

Reason: Using the same terminology and syntax facilitates understanding. Otherwise, users wonder whether some difference is implied by use of synonymous terms and variable syntax.

#### Be appropriate for the type of metadata item being defined

Explanation: Different types of metadata item in a metadata registry (e.g. data element concept, data element, conceptual domain, value domain) each play a different role and this should be reflected in the definitions.

Example: Data element concept: “Job Grade Maximum Salary Amount”

Definition: The maximum salary permitted for the associated job grade.

Note: The data element concept makes no reference to a specific value domain.

Conceptual Domain: “Monetary amount”

Definition: An amount that may be expressed in a unit of currency.

Note: The definition refers to a “dimensionality” of currency, but not to a specific currency.

Data element 1: “European Job Grade Maximum Salary Amount”

Definition: The maximum salary permitted for the associated job grade expressed in Euros.

Data element 2: “U.S. Job Grade Maximum Salary Amount”

Definition: The maximum salary permitted for the associated job grade expressed in US dollars.

Note: Data element definitions may refer to explicit values domains, since this may be all that distinguishes two data elements.

## Parts of Speech

### Nouns

Nouns name persons, places, things, ideas, or qualities, e.g., Franklin, boy, Yangtze River, shoreline, Bible, desk, fear, happiness.

### Pronouns

Pronouns usually substitute for nouns and function as nouns, e.g., I, you, he, she, it, we, they, myself, this, that, who, which, everyone.

### Verbs

Verbs express actions, occurrences, or states of being, e.g., be, become, inflate, and run.

### Adjectives

Adjectives describe or modify nouns or pronouns, e.g., gentle, helpful, small.

### Adverbs

Adverbs describe or modify verbs, adjectives, or other adverbs, e.g., almost, gently, helpfully, someday.

### Prepositions

Prepositions relate nouns or pronouns to other words in a sentence, e.g., about, at, down, for, of, with.

### Conjunctions

Conjunctions link words, clauses, and phrases. There are coordinating conjunctions that link words, clauses, or phrases of equal importance, and there are subordinating conjunctions that introduce subordinate clauses and link them to main clauses.

### Interjections

Interjections express feeling or command attention, either alone or in a sentence, e.g., darn, hey, oh, wow.

### Articles

Articles are base adjectives that are used to modify a noun, however their value is low, e.g. a, an, the.

### Quantifiers

Quantifiers provide information about the number or size of something, e.g. much, many, few, lots.

### Numerals

Numerals are specific numbers.

## Reserved Words

| **Reserved Word** | **ANSI** | **ODBC** | **MS SQL** | **MySql** | **Oracle** | **IBM DB2** | **PostgreSQL** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Absolute | Y | Y | Y | N | N | N | N |
| Access | N | N | N | N | Y | N | N |
| Accessible | N | N | N | Y | N | N | N |
| Account | N | N | N | N | Y | N | N |
| Action | Y | Y | Y | Y | N | N | N |
| Activate | N | N | N | N | Y | N | N |
| Ada | N | Y | N | N | N | N | N |
| Add | Y | Y | Y | Y | Y | Y | N |
| Admin | N | N | Y | N | Y | N | N |
| Advise | N | N | N | N | Y | N | N |
| After | N | N | Y | Y | Y | Y | N |
| Against | N | N | N | Y | N | N | N |
| Aggregate | N | N | Y | Y | N | N | N |
| Algorithm | N | N | N | Y | N | N | N |
| All | Y | Y | Y | Y | Y | Y | Y |
| All\_Rows | N | N | N | N | Y | N | N |
| Allocate | Y | Y | Y | N | Y | Y | N |
| Allow | N | N | N | N | N | Y | N |
| Alter | Y | Y | Y | Y | Y | Y | N |
| Analyse | N | N | N | N | N | N | Y |
| Analyze | N | N | N | Y | Y | N | Y |
| And | Y | Y | Y | Y | Y | Y | Y |
| Any | Y | Y | Y | Y | Y | Y | Y |
| Archive | N | N | N | N | Y | N | N |
| Archivelog | N | N | N | N | Y | N | N |
| Are | Y | Y | Y | N | N | N | N |
| Array | N | N | Y | N | Y | N | N |
| As | Y | Y | Y | Y | Y | Y | Y |
| Asc | Y | Y | Y | Y | Y | N | Y |
| Ascii | N | N | N | Y | N | N | N |
| Asensitive | N | N | Y | Y | N | Y | N |
| Assertion | Y | Y | Y | N | N | N | N |
| Associate | N | N | N | N | N | Y | N |
| Asutime | N | N | N | N | N | Y | N |
| At | Y | Y | Y | Y | Y | Y | N |
| Audit | N | N | N | N | Y | Y | N |
| Authenticated | N | N | N | N | Y | N | N |
| Authorization | Y | Y | Y | N | Y | N | Y |
| Authors | N | N | N | Y | N | N | N |
| Auto\_Increment | N | N | N | Y | N | N | N |
| Autoextend | N | N | N | N | Y | N | N |
| Autoextend\_Size | N | N | N | Y | N | N | N |
| Automatic | N | N | N | N | Y | N | N |
| Aux | N | N | N | N | N | Y | N |
| Auxiliary | N | N | N | N | N | Y | N |
| Avg | Y | Y | N | Y | N | N | N |
| Avg\_Row\_Length | N | N | N | Y | N | N | N |
| Backup | N | N | Y | Y | Y | N | N |
| Become | N | N | N | N | Y | N | N |
| Before | N | N | Y | Y | Y | Y | N |
| Begin | Y | Y | Y | Y | Y | Y | N |
| Between | Y | Y | Y | Y | Y | Y | Y |
| Bfile | N | N | N | N | Y | N | N |
| Bigint | N | N | N | Y | N | N | Y |
| Binary | N | N | Y | Y | N | N | Y |
| Binlog | N | N | N | Y | N | N | N |
| Bit | Y | Y | Y | Y | N | N | Y |
| Bit\_Length | Y | Y | N | N | N | N | N |
| Bitmap | N | N | N | N | Y | N | N |
| Blob | N | N | Y | Y | Y | N | N |
| Block | N | N | N | Y | Y | N | N |
| Body | N | N | N | N | Y | N | N |
| Bool | N | N | N | Y | N | N | N |
| Boolean | N | N | Y | Y | N | N | Y |
| Both | Y | Y | Y | Y | N | N | Y |
| Btree | N | N | N | Y | N | N | N |
| Bufferpool | N | N | N | N | N | Y | N |
| By | Y | Y | Y | Y | Y | Y | N |
| Byte | N | N | N | Y | N | N | N |
| Cache | N | N | N | Y | Y | N | N |
| Cache\_Instances | N | N | N | N | Y | N | N |
| Call | N | N | Y | Y | N | Y | N |
| Cancel | N | N | N | N | Y | N | N |
| Capture | N | N | N | N | N | Y | N |
| Cascade | Y | Y | Y | Y | Y | N | N |
| Cascaded | Y | Y | Y | Y | N | Y | N |
| Case | Y | Y | Y | Y | N | Y | Y |
| Cast | Y | Y | Y | N | Y | Y | Y |
| Catalog | Y | Y | Y | N | N | N | N |
| Catalog\_Name | N | N | N | Y | N | N | N |
| Ccsid | N | N | N | N | N | Y | N |
| Cfile | N | N | N | N | Y | N | N |
| Chain | N | N | N | Y | N | N | N |
| Chained | N | N | N | N | Y | N | N |
| Change | N | N | N | Y | Y | N | N |
| Changed | N | N | N | Y | N | N | N |
| Char | Y | Y | Y | Y | Y | Y | Y |
| Char\_Cs | N | N | N | N | Y | N | N |
| Char\_Length | Y | Y | N | N | N | N | N |
| Character | Y | Y | Y | Y | Y | Y | Y |
| Character\_Length | Y | Y | N | N | N | N | N |
| Charset | N | N | N | Y | N | N | N |
| Check | Y | Y | Y | Y | Y | Y | Y |
| Checkpoint | N | N | Y | N | Y | N | N |
| Checksum | N | N | N | Y | N | N | N |
| Choose | N | N | N | N | Y | N | N |
| Chunk | N | N | N | N | Y | N | N |
| Cipher | N | N | N | Y | N | N | N |
| Class\_Origin | N | N | N | Y | N | N | N |
| Clear | N | N | N | N | Y | N | N |
| Client | N | N | N | Y | N | N | N |
| Clob | N | N | Y | N | Y | N | N |
| Clone | N | N | N | N | Y | Y | N |
| Close | Y | Y | Y | Y | Y | Y | N |
| Close\_Cached\_Open\_Cursors | N | N | N | N | Y | N | N |
| Cluster | N | N | N | N | Y | Y | N |
| Coalesce | Y | Y | Y | Y | Y | N | Y |
| Code | N | N | N | Y | N | N | N |
| Collate | Y | Y | Y | Y | N | N | Y |
| Collation | Y | Y | Y | Y | N | N | N |
| Collection | N | N | N | N | N | Y | N |
| Collid | N | N | N | N | N | Y | N |
| Column | Y | Y | Y | Y | Y | Y | Y |
| Column\_Name | N | N | N | Y | N | N | N |
| Columns | N | N | N | Y | Y | N | N |
| Comment | N | N | N | Y | Y | Y | N |
| Commit | Y | Y | Y | Y | Y | Y | N |
| Committed | N | N | N | Y | Y | N | N |
| Compact | N | N | N | Y | N | N | N |
| Compatibility | N | N | N | N | Y | N | N |
| Compile | N | N | N | N | Y | N | N |
| Complete | N | N | N | N | Y | N | N |
| Completion | N | N | Y | Y | N | N | N |
| Composite\_Limit | N | N | N | N | Y | N | N |
| Compress | N | N | N | N | Y | N | N |
| Compressed | N | N | N | Y | N | N | N |
| Compute | N | N | Y | N | Y | N | N |
| Concat | N | N | N | N | N | Y | N |
| Concurrent | N | N | N | Y | N | N | N |
| Condition | N | N | Y | Y | N | Y | N |
| Connect | Y | Y | Y | N | Y | Y | N |
| Connect\_Time | N | N | N | N | Y | N | N |
| Connection | Y | Y | Y | Y | N | Y | N |
| Consistent | N | N | N | Y | N | N | N |
| Constraint | Y | Y | Y | Y | Y | Y | Y |
| Constraint\_Catalog | N | N | N | Y | N | N | N |
| Constraint\_Name | N | N | N | Y | N | N | N |
| Constraint\_Schema | N | N | N | Y | N | N | N |
| Constraints | Y | Y | Y | N | Y | N | N |
| Contains | N | N | Y | Y | N | Y | N |
| Content | N | N | N | N | N | Y | N |
| Contents | N | N | N | N | Y | N | N |
| Context | N | N | N | Y | N | N | N |
| Continue | Y | Y | Y | Y | Y | Y | N |
| Contributors | N | N | N | Y | N | N | N |
| Controlfile | N | N | N | N | Y | N | N |
| Convert | Y | Y | Y | Y | Y | N | Y |
| Corresponding | Y | Y | Y | N | N | N | N |
| Cost | N | N | N | N | Y | N | N |
| Count | Y | Y | N | N | N | N | N |
| Cpu | N | N | N | Y | N | N | N |
| Cpu\_Per\_Call | N | N | N | N | Y | N | N |
| Cpu\_Per\_Session | N | N | N | N | Y | N | N |
| Create | Y | Y | Y | Y | Y | Y | Y |
| Cross | Y | Y | Y | Y | N | N | Y |
| Cube | N | N | Y | Y | N | N | N |
| Curren\_User | N | N | N | N | Y | N | N |
| Current | Y | Y | Y | N | Y | Y | N |
| Current\_Date | Y | Y | Y | Y | N | Y | Y |
| Current\_Lc\_Ctype | N | N | N | N | N | Y | N |
| Current\_Path | N | N | Y | N | N | Y | N |
| Current\_Schema | N | N | Y | N | Y | Y | N |
| Current\_Time | Y | Y | Y | Y | N | Y | Y |
| Current\_Timestamp | Y | Y | Y | Y | N | Y | Y |
| Current\_User | Y | Y | Y | Y | N | N | Y |
| Currval | N | N | N | N | N | Y | N |
| Cursor | Y | Y | Y | Y | Y | Y | N |
| Cursor\_Name | N | N | N | Y | N | N | N |
| Cycle | N | N | Y | N | Y | N | N |
| Dangling | N | N | N | N | Y | N | N |
| Data | N | N | Y | Y | N | Y | N |
| Database | N | N | Y | Y | Y | Y | N |
| Databases | N | N | N | Y | N | N | N |
| Datafile | N | N | N | Y | Y | N | N |
| Datafiles | N | N | N | N | Y | N | N |
| Dataobjno | N | N | N | N | Y | N | N |
| Date | Y | Y | Y | Y | Y | N | N |
| Datetime | N | N | N | Y | N | N | N |
| Day | Y | Y | Y | Y | N | Y | N |
| Day\_Hour | N | N | N | Y | N | N | N |
| Day\_Microsecond | N | N | N | Y | N | N | N |
| Day\_Minute | N | N | N | Y | N | N | N |
| Day\_Second | N | N | N | Y | N | N | N |
| Days | N | N | N | N | N | Y | N |
| Dba | N | N | N | N | Y | N | N |
| Dbhigh | N | N | N | N | Y | N | N |
| Dbinfo | N | N | N | N | N | Y | N |
| Dblow | N | N | N | N | Y | N | N |
| Dbmac | N | N | N | N | Y | N | N |
| Deallocate | Y | Y | Y | Y | Y | N | N |
| Debug | N | N | N | N | Y | N | N |
| Dec | Y | Y | Y | Y | Y | N | Y |
| Decimal | Y | Y | Y | Y | Y | N | Y |
| Declare | Y | Y | Y | Y | Y | Y | N |
| Default | Y | Y | Y | Y | Y | Y | Y |
| Deferrable | Y | Y | Y | N | Y | N | Y |
| Deferred | Y | Y | Y | N | Y | N | N |
| Definer | N | N | N | Y | N | N | N |
| Degree | N | N | N | N | Y | N | N |
| Delay\_Key\_Write | N | N | N | Y | N | N | N |
| Delayed | N | N | N | Y | N | N | N |
| Delete | Y | Y | Y | Y | Y | Y | N |
| Deref | N | N | Y | N | Y | N | N |
| Des\_Key\_File | N | N | N | Y | N | N | N |
| Desc | Y | Y | Y | Y | Y | N | Y |
| Describe | Y | Y | Y | Y | N | N | N |
| Descriptor | Y | Y | Y | N | N | Y | N |
| Deterministic | N | N | Y | Y | N | Y | N |
| Diagnostics | Y | Y | Y | N | N | N | N |
| Directory | N | N | N | Y | Y | N | N |
| Disable | N | N | N | Y | Y | Y | N |
| Disallow | N | N | N | N | N | Y | N |
| Discard | N | N | N | Y | N | N | N |
| Disconnect | Y | Y | Y | N | Y | N | N |
| Disk | N | N | Y | Y | N | N | N |
| Dismount | N | N | N | N | Y | N | N |
| Distinct | Y | Y | Y | Y | Y | Y | Y |
| Distinctrow | N | N | N | Y | N | N | N |
| Distributed | N | N | Y | N | Y | N | N |
| Div | N | N | N | Y | N | N | N |
| Dml | N | N | N | N | Y | N | N |
| Do | N | N | N | Y | N | Y | Y |
| Document | N | N | N | N | N | Y | N |
| Domain | Y | Y | Y | N | N | N | N |
| Double | Y | Y | Y | Y | Y | Y | N |
| Drop | Y | Y | Y | Y | Y | Y | N |
| Dssize | N | N | N | N | N | Y | N |
| Dual | N | N | N | Y | N | N | N |
| Dump | N | N | Y | N | Y | N | N |
| Dumpfile | N | N | N | Y | N | N | N |
| Duplicate | N | N | N | Y | N | N | N |
| Dynamic | N | N | Y | Y | N | Y | N |
| Each | N | N | Y | Y | Y | N | N |
| Editproc | N | N | N | N | N | Y | N |
| Else | Y | Y | Y | Y | Y | Y | Y |
| Elseif | N | N | N | Y | N | Y | N |
| Enable | N | N | N | Y | Y | N | N |
| Enclosed | N | N | N | Y | N | N | N |
| Encoding | N | N | N | N | N | Y | N |
| Encryption | N | N | N | N | N | Y | N |
| End | Y | Y | Y | Y | Y | Y | Y |
| End-Exec | Y | Y | Y | N | N | Y | N |
| Ending | N | N | N | N | N | Y | N |
| Ends | N | N | N | Y | N | N | N |
| Enforce | N | N | N | N | Y | N | N |
| Engine | N | N | N | Y | N | N | N |
| Engines | N | N | N | Y | N | N | N |
| Entry | N | N | N | N | Y | N | N |
| Enum | N | N | N | Y | N | N | N |
| Erase | N | N | N | N | N | Y | N |
| Error | N | N | N | Y | N | N | N |
| Errors | N | N | N | Y | N | N | N |
| Escape | Y | Y | Y | Y | Y | Y | N |
| Escaped | N | N | N | Y | N | N | N |
| Event | N | N | N | Y | N | N | N |
| Events | N | N | N | Y | N | N | N |
| Every | N | N | Y | Y | N | N | N |
| Except | Y | Y | Y | N | Y | Y | Y |
| Exception | Y | Y | Y | N | N | Y | N |
| Exceptions | N | N | N | N | Y | N | N |
| Exchange | N | N | N | N | Y | N | N |
| Excluding | N | N | N | N | Y | N | N |
| Exclusive | N | N | N | N | Y | N | N |
| Exec | Y | Y | Y | N | N | N | N |
| Execute | Y | Y | Y | Y | Y | Y | N |
| Exists | Y | Y | Y | Y | Y | Y | Y |
| Exit | N | N | Y | Y | N | Y | N |
| Expansion | N | N | N | Y | N | N | N |
| Expire | N | N | N | N | Y | N | N |
| Explain | N | N | N | Y | Y | Y | N |
| Extended | N | N | N | Y | N | N | N |
| Extent | N | N | N | N | Y | N | N |
| Extent\_Size | N | N | N | Y | N | N | N |
| Extents | N | N | N | N | Y | N | N |
| External | Y | Y | Y | N | N | Y | N |
| Externally | N | N | N | N | Y | N | N |
| Extract | Y | Y | N | N | N | N | Y |
| Failed\_Login\_Attempts | N | N | N | N | Y | N | N |
| False | Y | Y | Y | Y | Y | N | Y |
| Fast | N | N | N | Y | Y | N | N |
| Faults | N | N | N | Y | N | N | N |
| Fenced | N | N | N | N | N | Y | N |
| Fetch | Y | Y | Y | Y | N | Y | N |
| Fieldproc | N | N | N | N | N | Y | N |
| Fields | N | N | N | Y | N | N | N |
| File | N | N | Y | Y | Y | N | N |
| Final | N | N | N | N | N | Y | N |
| First | Y | Y | Y | Y | N | Y | N |
| First\_Rows | N | N | N | N | Y | N | N |
| Fixed | N | N | N | Y | N | N | N |
| Flagger | N | N | N | N | Y | N | N |
| Float | Y | Y | Y | Y | Y | N | Y |
| Float4 | N | N | N | Y | N | N | N |
| Float8 | N | N | N | Y | N | N | N |
| Flob | N | N | N | N | Y | N | N |
| Flush | N | N | N | Y | Y | N | N |
| For | Y | Y | Y | Y | Y | Y | Y |
| Force | N | N | N | Y | Y | N | N |
| Foreign | Y | Y | Y | Y | Y | N | Y |
| Fortran | N | Y | N | N | N | N | N |
| Found | Y | Y | Y | Y | N | N | N |
| Frac\_Second | N | N | N | Y | N | N | N |
| Free | N | N | Y | N | N | Y | N |
| Freelist | N | N | N | N | Y | N | N |
| Freelists | N | N | N | N | Y | N | N |
| Freeze | N | N | N | N | N | N | Y |
| From | Y | Y | Y | Y | Y | Y | Y |
| Full | Y | Y | Y | Y | Y | Y | Y |
| Fulltext | N | N | N | Y | N | N | N |
| Function | N | N | Y | Y | Y | Y | N |
| General | N | N | Y | Y | N | N | N |
| Generated | N | N | N | N | N | Y | N |
| Geometry | N | N | N | Y | N | N | N |
| Geometrycollection | N | N | N | Y | N | N | N |
| Get | Y | Y | Y | N | N | Y | N |
| Get\_Format | N | N | N | Y | N | N | N |
| Global | Y | Y | Y | Y | Y | Y | N |
| Global\_Name | N | N | N | N | Y | N | N |
| Globally | N | N | N | N | Y | N | N |
| Go | Y | Y | Y | N | N | Y | N |
| Goto | Y | Y | Y | N | N | Y | N |
| Grant | Y | Y | Y | Y | Y | Y | Y |
| Grants | N | N | N | Y | N | N | N |
| Group | Y | Y | Y | Y | Y | Y | Y |
| Groups | N | N | N | N | Y | N | N |
| Handler | N | N | N | Y | N | Y | N |
| Hash | N | N | N | Y | Y | N | N |
| Hashkeys | N | N | N | N | Y | N | N |
| Having | Y | Y | Y | Y | Y | Y | Y |
| Header | N | N | N | N | Y | N | N |
| Heap | N | N | N | N | Y | N | N |
| Help | N | N | N | Y | N | N | N |
| High\_Priority | N | N | N | Y | N | N | N |
| Hold | N | N | Y | N | N | Y | N |
| Host | N | N | Y | Y | N | N | N |
| Hosts | N | N | N | Y | N | N | N |
| Hour | Y | Y | Y | Y | N | Y | N |
| Hour\_Microsecond | N | N | N | Y | N | N | N |
| Hour\_Minute | N | N | N | Y | N | N | N |
| Hour\_Second | N | N | N | Y | N | N | N |
| Hours | N | N | N | N | N | Y | N |
| Identified | N | N | N | Y | Y | N | N |
| Identity | Y | Y | Y | N | N | N | N |
| Idgenerators | N | N | N | N | Y | N | N |
| Idle\_Time | N | N | N | N | Y | N | N |
| If | N | N | Y | Y | Y | Y | N |
| Ignore | N | N | Y | Y | N | N | N |
| Ignore\_Server\_Ids | N | N | N | Y | N | N | N |
| Ilike | N | N | N | N | N | N | Y |
| Immediate | Y | Y | Y | N | Y | Y | N |
| Import | N | N | N | Y | N | N | N |
| In | Y | Y | Y | Y | Y | Y | Y |
| Include | N | Y | N | N | N | N | N |
| Including | N | N | N | N | Y | N | N |
| Inclusive | N | N | N | N | N | Y | N |
| Increment | N | N | N | N | Y | N | N |
| Ind\_Partition | N | N | N | N | Y | N | N |
| Index | N | Y | Y | Y | Y | Y | N |
| Indexed | N | N | N | N | Y | N | N |
| Indexes | N | N | N | Y | Y | N | N |
| Indicator | Y | Y | Y | N | Y | N | N |
| Infile | N | N | N | Y | N | N | N |
| Inherit | N | N | N | N | N | Y | N |
| Initial | N | N | N | N | Y | N | N |
| Initial\_Size | N | N | N | Y | N | N | N |
| Initially | Y | Y | Y | N | Y | N | Y |
| Initrans | N | N | N | N | Y | N | N |
| Inner | Y | Y | Y | Y | N | Y | Y |
| Innobase | N | N | N | Y | N | N | N |
| Innodb | N | N | N | Y | N | N | N |
| Inout | N | N | Y | Y | N | Y | N |
| Input | Y | Y | Y | N | N | N | N |
| Insensitive | Y | Y | N | Y | N | Y | N |
| Insert | Y | Y | Y | Y | Y | Y | N |
| Insert\_Method | N | N | N | Y | N | N | N |
| Install | N | N | N | Y | N | N | N |
| Instance | N | N | N | N | Y | N | N |
| Instances | N | N | N | N | Y | N | N |
| Instead | N | N | N | N | Y | N | N |
| Int | Y | Y | Y | Y | Y | N | Y |
| Int1 | N | N | N | Y | N | N | N |
| Int2 | N | N | N | Y | N | N | N |
| Int3 | N | N | N | Y | N | N | N |
| Int4 | N | N | N | Y | N | N | N |
| Int8 | N | N | N | Y | N | N | N |
| Integer | Y | Y | Y | Y | Y | N | Y |
| Intermediate | N | N | N | N | Y | N | N |
| Intersect | Y | Y | Y | N | Y | Y | Y |
| Interval | Y | Y | Y | Y | N | N | Y |
| Into | Y | Y | Y | Y | Y | Y | Y |
| Invoker | N | N | N | Y | N | N | N |
| Io | N | N | N | Y | N | N | N |
| Io\_Thread | N | N | N | Y | N | N | N |
| Ipc | N | N | N | Y | N | N | N |
| Is | Y | Y | Y | Y | Y | Y | Y |
| Isnull | N | N | N | N | N | N | Y |
| Isobid | N | N | N | N | N | Y | N |
| Isolation | Y | Y | Y | Y | Y | N | N |
| Isolation\_Level | N | N | N | N | Y | N | N |
| Issuer | N | N | N | Y | N | N | N |
| Iterate | N | N | Y | Y | N | Y | N |
| Jar | N | N | N | N | N | Y | N |
| Join | Y | Y | Y | Y | N | Y | Y |
| Keep | N | N | N | N | Y | Y | N |
| Key | Y | Y | Y | Y | Y | Y | N |
| Key\_Block\_Size | N | N | N | Y | N | N | N |
| Keys | N | N | N | Y | N | N | N |
| Kill | N | N | Y | Y | Y | N | N |
| Label | N | N | N | N | Y | Y | N |
| Language | Y | Y | Y | Y | N | Y | N |
| Last | Y | Y | Y | Y | N | Y | N |
| Layer | N | N | N | N | Y | N | N |
| Lc\_Ctype | N | N | N | N | N | Y | N |
| Leading | Y | Y | Y | Y | N | N | Y |
| Leave | N | N | N | Y | N | Y | N |
| Leaves | N | N | N | Y | N | N | N |
| Left | Y | Y | Y | Y | N | Y | Y |
| Less | N | N | Y | Y | Y | N | N |
| Level | Y | Y | Y | Y | Y | N | N |
| Library | N | N | N | N | Y | N | N |
| Like | Y | Y | Y | Y | Y | Y | Y |
| Limit | N | N | Y | Y | Y | N | Y |
| Linear | N | N | N | Y | N | N | N |
| Lines | N | N | N | Y | N | N | N |
| Linestring | N | N | N | Y | N | N | N |
| Link | N | N | N | N | Y | N | N |
| List | N | N | N | Y | Y | N | N |
| Load | N | N | Y | Y | N | N | N |
| Lob | N | N | N | N | Y | N | N |
| Local | Y | Y | Y | Y | Y | Y | N |
| Locale | N | N | N | N | N | Y | N |
| Localtime | N | N | Y | Y | N | N | Y |
| Localtimestamp | N | N | Y | Y | N | N | Y |
| Locator | N | N | Y | N | N | Y | N |
| Locators | N | N | N | N | N | Y | N |
| Lock | N | N | N | Y | Y | Y | N |
| Locked | N | N | N | N | Y | N | N |
| Lockmax | N | N | N | N | N | Y | N |
| Locks | N | N | N | Y | N | N | N |
| Locksize | N | N | N | N | N | Y | N |
| Log | N | N | N | N | Y | N | N |
| Logfile | N | N | N | Y | Y | N | N |
| Logging | N | N | N | N | Y | N | N |
| Logical\_Reads\_Per\_Call | N | N | N | N | Y | N | N |
| Logical\_Reads\_Per\_Session | N | N | N | N | Y | N | N |
| Logs | N | N | N | Y | N | N | N |
| Long | N | N | N | Y | Y | Y | N |
| Longblob | N | N | N | Y | N | N | N |
| Longtext | N | N | N | Y | N | N | N |
| Loop | N | N | N | Y | N | Y | N |
| Low\_Priority | N | N | N | Y | N | N | N |
| Lower | Y | Y | N | N | N | N | N |
| Maintained | N | N | N | N | N | Y | N |
| Manage | N | N | N | N | Y | N | N |
| Master | N | N | N | Y | Y | N | N |
| Master\_Connect\_Retry | N | N | N | Y | N | N | N |
| Master\_Heartbeat\_Period | N | N | N | Y | N | N | N |
| Master\_Host | N | N | N | Y | N | N | N |
| Master\_Log\_File | N | N | N | Y | N | N | N |
| Master\_Log\_Pos | N | N | N | Y | N | N | N |
| Master\_Password | N | N | N | Y | N | N | N |
| Master\_Port | N | N | N | Y | N | N | N |
| Master\_Server\_Id | N | N | N | Y | N | N | N |
| Master\_Ssl | N | N | N | Y | N | N | N |
| Master\_Ssl\_Ca | N | N | N | Y | N | N | N |
| Master\_Ssl\_Capath | N | N | N | Y | N | N | N |
| Master\_Ssl\_Cert | N | N | N | Y | N | N | N |
| Master\_Ssl\_Cipher | N | N | N | Y | N | N | N |
| Master\_Ssl\_Key | N | N | N | Y | N | N | N |
| Master\_Ssl\_Verify\_Server\_Cert | N | N | N | Y | N | N | N |
| Master\_User | N | N | N | Y | N | N | N |
| Match | Y | Y | Y | Y | N | N | N |
| Materialized | N | N | N | N | N | Y | N |
| Max | Y | Y | N | N | Y | N | N |
| Max\_Connections\_Per\_Hour | N | N | N | Y | N | N | N |
| Max\_Queries\_Per\_Hour | N | N | N | Y | N | N | N |
| Max\_Rows | N | N | N | Y | N | N | N |
| Max\_Size | N | N | N | Y | N | N | N |
| Max\_Updates\_Per\_Hour | N | N | N | Y | N | N | N |
| Max\_User\_Connections | N | N | N | Y | N | N | N |
| Maxarchlogs | N | N | N | N | Y | N | N |
| Maxdatafiles | N | N | N | N | Y | N | N |
| Maxextents | N | N | N | N | Y | N | N |
| Maxinstances | N | N | N | N | Y | N | N |
| Maxlogfiles | N | N | N | N | Y | N | N |
| Maxloghistory | N | N | N | N | Y | N | N |
| Maxlogmembers | N | N | N | N | Y | N | N |
| Maxsize | N | N | N | N | Y | N | N |
| Maxtrans | N | N | N | N | Y | N | N |
| Maxvalue | N | N | N | Y | Y | N | N |
| Medium | N | N | N | Y | N | N | N |
| Mediumblob | N | N | N | Y | N | N | N |
| Mediumint | N | N | N | Y | N | N | N |
| Mediumtext | N | N | N | Y | N | N | N |
| Member | N | N | Y | N | Y | N | N |
| Memory | N | N | N | Y | N | N | N |
| Merge | N | N | Y | Y | N | N | N |
| Message\_Text | N | N | N | Y | N | N | N |
| Microsecond | N | N | N | Y | N | Y | N |
| Microseconds | N | N | N | N | N | Y | N |
| Middleint | N | N | N | Y | N | N | N |
| Migrate | N | N | N | Y | N | N | N |
| Min | Y | Y | N | N | Y | N | N |
| Min\_Rows | N | N | N | Y | N | N | N |
| Minextents | N | N | N | N | Y | N | N |
| Minimum | N | N | N | N | Y | N | N |
| Minus | N | N | N | N | Y | N | N |
| Minute | Y | Y | Y | Y | N | Y | N |
| Minute\_Microsecond | N | N | N | Y | N | N | N |
| Minute\_Second | N | N | N | Y | N | N | N |
| Minutes | N | N | N | N | N | Y | N |
| Minvalue | N | N | N | N | Y | N | N |
| Mls\_Label\_Format | N | N | N | N | Y | N | N |
| Mlslabel | N | N | N | N | Y | N | N |
| Mod | N | N | Y | Y | N | N | N |
| Mode | N | N | N | Y | Y | N | N |
| Modifies | N | N | Y | Y | N | Y | N |
| Modify | N | N | Y | Y | Y | N | N |
| Module | Y | Y | Y | N | N | N | N |
| Month | Y | Y | Y | Y | N | Y | N |
| Months | N | N | N | N | N | Y | N |
| Mount | N | N | N | N | Y | N | N |
| Move | N | N | N | N | Y | N | N |
| Mts\_Dispatchers | N | N | N | N | Y | N | N |
| Multilinestring | N | N | N | Y | N | N | N |
| Multipoint | N | N | N | Y | N | N | N |
| Multipolygon | N | N | N | Y | N | N | N |
| Multiset | N | N | Y | N | Y | N | N |
| Mutex | N | N | N | Y | N | N | N |
| Mysql\_Errno | N | N | N | Y | N | N | N |
| Name | N | N | N | Y | N | N | N |
| Names | Y | Y | Y | Y | N | N | N |
| National | Y | Y | Y | Y | Y | N | N |
| Natural | Y | Y | Y | Y | N | N | Y |
| Nchar | Y | Y | Y | Y | Y | N | Y |
| Nchar\_Cs | N | N | N | N | Y | N | N |
| Nclob | N | N | Y | N | Y | N | N |
| Ndb | N | N | N | Y | N | N | N |
| Ndbcluster | N | N | N | Y | N | N | N |
| Needed | N | N | N | N | Y | N | N |
| Nested | N | N | N | N | Y | N | N |
| Network | N | N | N | N | Y | N | N |
| New | N | N | Y | Y | Y | N | Y |
| Next | Y | Y | Y | Y | Y | Y | N |
| Nextval | N | N | N | N | N | Y | N |
| No | Y | Y | Y | Y | N | Y | N |
| No\_Wait | N | N | N | Y | N | N | N |
| No\_Write\_To\_Binlog | N | N | N | Y | N | N | N |
| Noarchivelog | N | N | N | N | Y | N | N |
| Noaudit | N | N | N | N | Y | N | N |
| Nocache | N | N | N | N | Y | N | N |
| Nocompress | N | N | N | N | Y | N | N |
| Nocycle | N | N | N | N | Y | N | N |
| Nodegroup | N | N | N | Y | N | N | N |
| Noforce | N | N | N | N | Y | N | N |
| Nologging | N | N | N | N | Y | N | N |
| Nomaxvalue | N | N | N | N | Y | N | N |
| Nominvalue | N | N | N | N | Y | N | N |
| None | N | Y | Y | Y | Y | Y | Y |
| Noorder | N | N | N | N | Y | N | N |
| Nooverride | N | N | N | N | Y | N | N |
| Noparallel | N | N | N | N | Y | N | N |
| Noreverse | N | N | N | N | Y | N | N |
| Normal | N | N | N | N | Y | N | N |
| Nosort | N | N | N | N | Y | N | N |
| Not | Y | Y | Y | Y | Y | Y | Y |
| Nothing | N | N | N | N | Y | N | N |
| Notnull | N | N | N | N | N | N | Y |
| Nowait | N | N | N | N | Y | N | N |
| Null | Y | Y | Y | Y | Y | Y | Y |
| Nullif | Y | Y | Y | N | N | N | Y |
| Nulls | N | N | N | N | N | Y | N |
| Number | N | N | N | N | Y | N | N |
| Numeric | Y | Y | Y | Y | Y | N | Y |
| Numparts | N | N | N | N | N | Y | N |
| Nvarchar | N | N | N | Y | N | N | N |
| Nvarchar2 | N | N | N | N | Y | N | N |
| Obid | N | N | N | N | N | Y | N |
| Object | N | N | Y | N | Y | N | N |
| Objno | N | N | N | N | Y | N | N |
| Objno\_Reuse | N | N | N | N | Y | N | N |
| Octet\_Length | Y | Y | N | N | N | N | N |
| Of | Y | Y | Y | N | Y | Y | N |
| Off | N | N | Y | N | Y | N | Y |
| Offline | N | N | N | N | Y | N | N |
| Offset | N | N | N | Y | N | N | Y |
| Oid | N | N | N | N | Y | N | N |
| Oidindex | N | N | N | N | Y | N | N |
| Old | N | N | Y | N | Y | Y | Y |
| Old\_Password | N | N | N | Y | N | N | N |
| On | Y | Y | Y | Y | Y | Y | Y |
| One | N | N | N | Y | N | N | N |
| One\_Shot | N | N | N | Y | N | N | N |
| Online | N | N | N | N | Y | N | N |
| Only | Y | Y | Y | N | Y | N | Y |
| Opcode | N | N | N | N | Y | N | N |
| Open | Y | Y | Y | Y | Y | Y | N |
| Optimal | N | N | N | N | Y | N | N |
| Optimization | N | N | N | N | N | Y | N |
| Optimize | N | N | N | Y | N | Y | N |
| Optimizer\_Goal | N | N | N | N | Y | N | N |
| Option | Y | Y | Y | Y | Y | N | N |
| Optionally | N | N | N | Y | N | N | N |
| Options | N | N | N | Y | N | N | N |
| Or | Y | Y | Y | Y | Y | Y | Y |
| Order | Y | Y | Y | Y | Y | Y | Y |
| Organization | N | N | N | N | Y | Y | N |
| Oslabel | N | N | N | N | Y | N | N |
| Out | N | N | Y | Y | N | Y | N |
| Outer | Y | Y | Y | Y | N | Y | Y |
| Outfile | N | N | N | Y | N | N | N |
| Output | Y | Y | Y | N | N | N | N |
| Overflow | N | N | N | N | Y | N | N |
| Overlaps | Y | Y | N | N | N | N | Y |
| Overlay | N | N | Y | N | N | N | Y |
| Own | N | N | N | N | Y | N | N |
| Owner | N | N | N | Y | N | N | N |
| Pack\_Keys | N | N | N | Y | N | N | N |
| Package | N | N | N | N | Y | Y | N |
| Pad | Y | Y | Y | N | N | N | N |
| Padded | N | N | N | N | N | Y | N |
| Page | N | N | N | Y | N | N | N |
| Parallel | N | N | N | N | Y | N | N |
| Parameter | N | N | Y | N | N | Y | N |
| Parser | N | N | N | Y | N | N | N |
| Part | N | N | N | N | N | Y | N |
| Partial | Y | Y | Y | Y | N | N | N |
| Partition | N | N | Y | Y | Y | Y | N |
| Partitioned | N | N | N | N | N | Y | N |
| Partitioning | N | N | N | Y | N | Y | N |
| Partitions | N | N | N | Y | N | N | N |
| Pascal | N | Y | N | N | N | N | N |
| Password | N | N | N | Y | Y | N | N |
| Password\_Grace\_Time | N | N | N | N | Y | N | N |
| Password\_Life\_Time | N | N | N | N | Y | N | N |
| Password\_Lock\_Time | N | N | N | N | Y | N | N |
| Password\_Reuse\_Max | N | N | N | N | Y | N | N |
| Password\_Reuse\_Time | N | N | N | N | Y | N | N |
| Password\_Verify\_Function | N | N | N | N | Y | N | N |
| Path | N | N | Y | N | N | Y | N |
| Pctfree | N | N | N | N | Y | N | N |
| Pctincrease | N | N | N | N | Y | N | N |
| Pctthreshold | N | N | N | N | Y | N | N |
| Pctused | N | N | N | N | Y | N | N |
| Pctversion | N | N | N | N | Y | N | N |
| Percent | N | N | Y | N | Y | N | N |
| Period | N | N | N | N | N | Y | N |
| Permanent | N | N | N | N | Y | N | N |
| Phase | N | N | N | Y | N | N | N |
| Piecesize | N | N | N | N | N | Y | N |
| Placing | N | N | N | N | N | N | Y |
| Plan | N | N | Y | N | Y | Y | N |
| Plsql\_Debug | N | N | N | N | Y | N | N |
| Plugin | N | N | N | Y | N | N | N |
| Plugins | N | N | N | Y | N | N | N |
| Point | N | N | N | Y | N | N | N |
| Polygon | N | N | N | Y | N | N | N |
| Port | N | N | N | Y | N | N | N |
| Position | Y | Y | N | N | N | N | Y |
| Post\_Transaction | N | N | N | N | Y | N | N |
| Precision | Y | Y | Y | Y | Y | Y | N |
| Prepare | Y | Y | Y | Y | N | Y | N |
| Preserve | Y | Y | Y | Y | Y | N | N |
| Prev | N | N | N | Y | N | N | N |
| Prevval | N | N | N | N | N | Y | N |
| Primary | Y | Y | Y | Y | Y | N | Y |
| Prior | Y | Y | Y | N | Y | Y | N |
| Priqty | N | N | N | N | N | Y | N |
| Private | N | N | N | N | Y | N | N |
| Private\_Sga | N | N | N | N | Y | N | N |
| Privilege | N | N | N | N | Y | N | N |
| Privileges | Y | Y | Y | Y | Y | Y | N |
| Procedure | Y | Y | Y | Y | Y | Y | N |
| Processlist | N | N | N | Y | N | N | N |
| Profile | N | N | N | Y | Y | N | N |
| Profiles | N | N | N | Y | N | N | N |
| Program | N | N | N | N | N | Y | N |
| Proxy | N | N | N | Y | N | N | N |
| Psid | N | N | N | N | N | Y | N |
| Public | Y | Y | Y | N | Y | Y | N |
| Purge | N | N | N | Y | Y | N | N |
| Quarter | N | N | N | Y | N | N | N |
| Query | N | N | N | Y | N | Y | N |
| Queryno | N | N | N | N | N | Y | N |
| Queue | N | N | N | N | Y | N | N |
| Quick | N | N | N | Y | N | N | N |
| Quota | N | N | N | N | Y | N | N |
| Range | N | N | Y | Y | Y | N | N |
| Raw | N | N | N | N | Y | N | N |
| Rba | N | N | N | N | Y | N | N |
| Read | Y | Y | Y | Y | Y | N | N |
| Read\_Only | N | N | N | Y | N | N | N |
| Read\_Write | N | N | N | Y | N | N | N |
| Reads | N | N | Y | Y | N | Y | N |
| Readup | N | N | N | N | Y | N | N |
| Real | Y | Y | Y | Y | Y | N | Y |
| Rebuild | N | N | N | Y | Y | N | N |
| Recover | N | N | N | Y | Y | N | N |
| Recoverable | N | N | N | N | Y | N | N |
| Recovery | N | N | N | N | Y | N | N |
| Redo\_Buffer\_Size | N | N | N | Y | N | N | N |
| Redofile | N | N | N | Y | N | N | N |
| Redundant | N | N | N | Y | N | N | N |
| Ref | N | N | Y | N | Y | N | N |
| References | Y | Y | Y | Y | Y | Y | Y |
| Referencing | N | N | Y | N | Y | N | N |
| Refresh | N | N | N | N | Y | Y | N |
| Regexp | N | N | N | Y | N | N | N |
| Relative | Y | Y | Y | N | N | N | N |
| Relay | N | N | N | Y | N | N | N |
| Relay\_Log\_File | N | N | N | Y | N | N | N |
| Relay\_Log\_Pos | N | N | N | Y | N | N | N |
| Relay\_Thread | N | N | N | Y | N | N | N |
| Relaylog | N | N | N | Y | N | N | N |
| Release | N | N | Y | Y | N | Y | N |
| Reload | N | N | N | Y | N | N | N |
| Remove | N | N | N | Y | N | N | N |
| Rename | N | N | N | Y | Y | Y | N |
| Reorganize | N | N | N | Y | N | N | N |
| Repair | N | N | N | Y | N | N | N |
| Repeat | N | N | N | Y | N | Y | N |
| Repeatable | N | N | N | Y | N | N | N |
| Replace | N | N | N | Y | Y | N | N |
| Replication | N | N | Y | Y | N | N | N |
| Require | N | N | N | Y | N | N | N |
| Reset | N | N | N | Y | Y | N | N |
| Resetlogs | N | N | N | N | Y | N | N |
| Resignal | N | N | N | Y | N | Y | N |
| Resize | N | N | N | N | Y | N | N |
| Resource | N | N | N | N | Y | N | N |
| Restore | N | N | Y | Y | N | N | N |
| Restrict | Y | Y | Y | Y | N | Y | N |
| Restricted | N | N | N | N | Y | N | N |
| Result | N | N | Y | N | N | Y | N |
| Result\_Set\_Locator | N | N | N | N | N | Y | N |
| Resume | N | N | N | Y | N | N | N |
| Return | N | N | Y | Y | Y | Y | N |
| Returning | N | N | N | N | Y | N | N |
| Returns | N | N | Y | Y | N | Y | N |
| Reuse | N | N | N | N | Y | N | N |
| Reverse | N | N | N | N | Y | N | N |
| Revoke | Y | Y | Y | Y | Y | Y | N |
| Right | Y | Y | Y | Y | N | Y | Y |
| Rlike | N | N | N | Y | N | N | N |
| Role | N | N | Y | N | Y | Y | N |
| Roles | N | N | N | N | Y | N | N |
| Rollback | Y | Y | Y | Y | Y | Y | N |
| Rollup | N | N | Y | Y | N | N | N |
| Round\_Ceiling | N | N | N | N | N | Y | N |
| Round\_Down | N | N | N | N | N | Y | N |
| Round\_Floor | N | N | N | N | N | Y | N |
| Round\_Half\_Down | N | N | N | N | N | Y | N |
| Round\_Half\_Even | N | N | N | N | N | Y | N |
| Round\_Half\_Up | N | N | N | N | N | Y | N |
| Round\_Up | N | N | N | N | N | Y | N |
| Routine | N | N | Y | Y | N | N | N |
| Row | N | N | Y | Y | Y | Y | Y |
| Row\_Format | N | N | N | Y | N | N | N |
| Rowid | N | N | N | N | Y | N | N |
| Rownum | N | N | N | N | Y | N | N |
| Rows | Y | Y | Y | Y | Y | N | N |
| Rowset | N | N | N | N | N | Y | N |
| Rtree | N | N | N | Y | N | N | N |
| Rule | N | N | Y | N | Y | N | N |
| Run | N | N | N | N | N | Y | N |
| Sample | N | N | N | N | Y | N | N |
| Savepoint | N | N | Y | Y | Y | Y | N |
| Sb4 | N | N | N | N | Y | N | N |
| Scan\_Instances | N | N | N | N | Y | N | N |
| Schedule | N | N | N | Y | N | N | N |
| Schema | Y | Y | Y | Y | Y | Y | N |
| Schema\_Name | N | N | N | Y | N | N | N |
| Schemas | N | N | N | Y | N | N | N |
| Scn | N | N | N | N | Y | N | N |
| Scope | N | N | Y | N | Y | N | N |
| Scratchpad | N | N | N | N | N | Y | N |
| Scroll | Y | Y | Y | N | N | N | N |
| Sd\_All | N | N | N | N | Y | N | N |
| Sd\_Inhibit | N | N | N | N | Y | N | N |
| Sd\_Show | N | N | N | N | Y | N | N |
| Second | Y | Y | Y | Y | N | Y | N |
| Second\_Microsecond | N | N | N | Y | N | N | N |
| Seconds | N | N | N | N | N | Y | N |
| Secqty | N | N | N | N | N | Y | N |
| Section | Y | Y | Y | N | N | N | N |
| Security | N | N | N | Y | N | Y | N |
| Seg\_Block | N | N | N | N | Y | N | N |
| Seg\_File | N | N | N | N | Y | N | N |
| Segment | N | N | N | N | Y | N | N |
| Select | Y | Y | Y | Y | Y | Y | Y |
| Sensitive | N | N | Y | Y | N | Y | N |
| Separator | N | N | N | Y | N | N | N |
| Sequence | N | N | Y | N | Y | Y | N |
| Serial | N | N | N | Y | N | N | N |
| Serializable | N | N | N | Y | Y | N | N |
| Server | N | N | N | Y | N | N | N |
| Session | Y | Y | Y | Y | Y | N | N |
| Session\_Cached\_Cursors | N | N | N | N | Y | N | N |
| Session\_User | Y | Y | Y | N | N | Y | Y |
| Sessions\_Per\_User | N | N | N | N | Y | N | N |
| Set | Y | Y | Y | Y | Y | Y | N |
| Setof | N | N | N | N | N | N | Y |
| Share | N | N | N | Y | Y | N | N |
| Shared | N | N | N | N | Y | N | N |
| Shared\_Pool | N | N | N | N | Y | N | N |
| Show | N | N | N | Y | N | N | N |
| Shrink | N | N | N | N | Y | N | N |
| Shutdown | N | N | Y | Y | N | N | N |
| Signal | N | N | N | Y | N | Y | N |
| Signed | N | N | N | Y | N | N | N |
| Similar | N | N | Y | N | N | N | Y |
| Simple | N | N | N | Y | N | Y | N |
| Size | Y | Y | Y | N | Y | N | N |
| Skip | N | N | N | N | Y | N | N |
| Skip\_Unusable\_Indexes | N | N | N | N | Y | N | N |
| Slave | N | N | N | Y | N | N | N |
| Slow | N | N | N | Y | N | N | N |
| Smallint | Y | Y | Y | Y | Y | N | Y |
| Snapshot | N | N | N | Y | Y | N | N |
| Socket | N | N | N | Y | N | N | N |
| Some | Y | Y | Y | Y | Y | Y | Y |
| Soname | N | N | N | Y | N | N | N |
| Sort | N | N | N | N | Y | N | N |
| Sounds | N | N | N | Y | N | N | N |
| Source | N | N | N | Y | N | Y | N |
| Space | Y | Y | Y | N | N | N | N |
| Spatial | N | N | N | Y | N | N | N |
| Specific | N | N | Y | Y | N | Y | N |
| Specification | N | N | N | N | Y | N | N |
| Split | N | N | N | N | Y | N | N |
| Sql | Y | Y | Y | Y | N | N | N |
| Sql\_Big\_Result | N | N | N | Y | N | N | N |
| Sql\_Buffer\_Result | N | N | N | Y | N | N | N |
| Sql\_Cache | N | N | N | Y | N | N | N |
| Sql\_Calc\_Found\_Rows | N | N | N | Y | N | N | N |
| Sql\_No\_Cache | N | N | N | Y | N | N | N |
| Sql\_Small\_Result | N | N | N | Y | N | N | N |
| Sql\_Thread | N | N | N | Y | N | N | N |
| Sql\_Trace | N | N | N | N | Y | N | N |
| Sql\_Tsi\_Day | N | N | N | Y | N | N | N |
| Sql\_Tsi\_Frac\_Second | N | N | N | Y | N | N | N |
| Sql\_Tsi\_Hour | N | N | N | Y | N | N | N |
| Sql\_Tsi\_Minute | N | N | N | Y | N | N | N |
| Sql\_Tsi\_Month | N | N | N | Y | N | N | N |
| Sql\_Tsi\_Quarter | N | N | N | Y | N | N | N |
| Sql\_Tsi\_Second | N | N | N | Y | N | N | N |
| Sql\_Tsi\_Week | N | N | N | Y | N | N | N |
| Sql\_Tsi\_Year | N | N | N | Y | N | N | N |
| Sqlca | N | Y | N | N | N | N | N |
| Sqlcode | Y | Y | N | N | N | N | N |
| Sqlerror | Y | Y | N | N | N | N | N |
| Sqlexception | N | N | Y | Y | N | N | N |
| Sqlstate | Y | Y | Y | Y | N | N | N |
| Sqlwarning | N | Y | Y | Y | N | N | N |
| Ssl | N | N | N | Y | N | N | N |
| Standard | N | N | N | N | N | Y | N |
| Standby | N | N | N | N | Y | N | N |
| Start | N | N | Y | Y | Y | N | N |
| Starting | N | N | N | Y | N | N | N |
| Starts | N | N | N | Y | N | N | N |
| Statement | N | N | Y | N | N | Y | N |
| Statement\_Id | N | N | N | N | Y | N | N |
| Static | N | N | Y | N | N | Y | N |
| Statistics | N | N | Y | N | Y | N | N |
| Status | N | N | N | Y | N | N | N |
| Stay | N | N | N | N | N | Y | N |
| Stogroup | N | N | N | N | N | Y | N |
| Stop | N | N | N | Y | Y | N | N |
| Storage | N | N | N | Y | Y | N | N |
| Store | N | N | N | N | Y | N | N |
| Stores | N | N | N | N | N | Y | N |
| Straight\_Join | N | N | N | Y | N | N | N |
| String | N | N | N | Y | N | N | N |
| Structure | N | N | Y | N | Y | N | N |
| Style | N | N | N | N | N | Y | N |
| Subclass\_Origin | N | N | N | Y | N | N | N |
| Subject | N | N | N | Y | N | N | N |
| Subpartition | N | N | N | Y | N | N | N |
| Subpartitions | N | N | N | Y | N | N | N |
| Substring | Y | Y | N | N | N | N | Y |
| Successful | N | N | N | N | Y | N | N |
| Sum | Y | Y | N | N | N | N | N |
| Summary | N | N | N | N | N | Y | N |
| Super | N | N | N | Y | N | N | N |
| Suspend | N | N | N | Y | N | N | N |
| Swaps | N | N | N | Y | N | N | N |
| Switch | N | N | N | N | Y | N | N |
| Switches | N | N | N | Y | N | N | N |
| Synonym | N | N | N | N | Y | Y | N |
| Sys\_Op\_Enforce\_Not\_Null$ | N | N | N | N | Y | N | N |
| Sys\_Op\_Ntcimg$ | N | N | N | N | Y | N | N |
| Sysdate | N | N | N | N | Y | Y | N |
| Sysdba | N | N | N | N | Y | N | N |
| Sysoper | N | N | N | N | Y | N | N |
| System | N | N | Y | N | Y | Y | N |
| System\_User | Y | Y | Y | N | N | N | N |
| Systimestamp | N | N | N | N | N | Y | N |
| Table | Y | Y | Y | Y | Y | Y | Y |
| Table\_Checksum | N | N | N | Y | N | N | N |
| Table\_Name | N | N | N | Y | N | N | N |
| Tables | N | N | N | Y | Y | N | N |
| Tablespace | N | N | N | Y | Y | Y | N |
| Tablespace\_No | N | N | N | N | Y | N | N |
| Tabno | N | N | N | N | Y | N | N |
| Temporary | Y | Y | Y | Y | Y | N | N |
| Temptable | N | N | N | Y | N | N | N |
| Terminated | N | N | N | Y | N | N | N |
| Text | N | N | N | Y | N | N | N |
| Than | N | N | Y | Y | Y | N | N |
| The | N | N | N | N | Y | N | N |
| Then | Y | Y | Y | Y | Y | Y | Y |
| Thread | N | N | N | N | Y | N | N |
| Time | Y | Y | Y | Y | Y | N | Y |
| Timestamp | Y | Y | Y | Y | Y | N | Y |
| Timestampadd | N | N | N | Y | N | N | N |
| Timestampdiff | N | N | N | Y | N | N | N |
| Timezone\_Hour | Y | Y | Y | N | N | N | N |
| Timezone\_Minute | Y | Y | Y | N | N | N | N |
| Tinyblob | N | N | N | Y | N | N | N |
| Tinyint | N | N | N | Y | N | N | N |
| Tinytext | N | N | N | Y | N | N | N |
| To | Y | Y | Y | Y | Y | Y | Y |
| Toplevel | N | N | N | N | Y | N | N |
| Trace | N | N | N | N | Y | N | N |
| Tracing | N | N | N | N | Y | N | N |
| Trailing | Y | Y | Y | Y | N | N | Y |
| Transaction | Y | Y | Y | Y | Y | N | N |
| Transitional | N | N | N | N | Y | N | N |
| Translate | Y | Y | N | N | N | N | N |
| Translation | Y | Y | Y | N | N | N | N |
| Treat | N | N | Y | N | N | N | Y |
| Trigger | N | N | Y | Y | Y | Y | N |
| Triggers | N | N | N | Y | Y | N | N |
| Trim | Y | Y | N | N | N | N | Y |
| True | Y | Y | Y | Y | Y | N | Y |
| Truncate | N | N | Y | Y | Y | Y | N |
| Tx | N | N | N | N | Y | N | N |
| Type | N | N | N | Y | Y | Y | N |
| Types | N | N | N | Y | N | N | N |
| Ub2 | N | N | N | N | Y | N | N |
| Uba | N | N | N | N | Y | N | N |
| Uid | N | N | N | N | Y | N | N |
| Unarchived | N | N | N | N | Y | N | N |
| Uncommitted | N | N | N | Y | N | N | N |
| Undefined | N | N | N | Y | N | N | N |
| Undo | N | N | N | Y | Y | Y | N |
| Undo\_Buffer\_Size | N | N | N | Y | N | N | N |
| Undofile | N | N | N | Y | N | N | N |
| Unicode | N | N | N | Y | N | N | N |
| Uninstall | N | N | N | Y | N | N | N |
| Union | Y | Y | Y | Y | Y | Y | Y |
| Unique | Y | Y | Y | Y | Y | Y | Y |
| Unknown | Y | Y | Y | Y | N | N | N |
| Unlimited | N | N | N | N | Y | N | N |
| Unlock | N | N | N | Y | Y | N | N |
| Unrecoverable | N | N | N | N | Y | N | N |
| Unsigned | N | N | N | Y | N | N | N |
| Until | N | N | N | Y | Y | Y | N |
| Unusable | N | N | N | N | Y | N | N |
| Unused | N | N | N | N | Y | N | N |
| Updatable | N | N | N | N | Y | N | N |
| Update | Y | Y | Y | Y | Y | Y | N |
| Upgrade | N | N | N | Y | N | N | N |
| Upper | Y | Y | N | N | N | N | N |
| Usage | Y | Y | Y | Y | Y | N | N |
| Use | N | N | Y | Y | Y | N | N |
| Use\_Frm | N | N | N | Y | N | N | N |
| User | Y | Y | Y | Y | Y | Y | Y |
| User\_Resources | N | N | N | Y | N | N | N |
| Using | Y | Y | Y | Y | Y | Y | Y |
| Utc\_Date | N | N | N | Y | N | N | N |
| Utc\_Time | N | N | N | Y | N | N | N |
| Utc\_Timestamp | N | N | N | Y | N | N | N |
| Validate | N | N | N | N | Y | N | N |
| Validation | N | N | N | N | Y | N | N |
| Validproc | N | N | N | N | N | Y | N |
| Value | Y | Y | Y | Y | Y | Y | N |
| Values | Y | Y | Y | Y | Y | Y | N |
| Varbinary | N | N | N | Y | N | N | N |
| Varchar | Y | Y | Y | Y | Y | N | Y |
| Varchar2 | N | N | N | N | Y | N | N |
| Varcharacter | N | N | N | Y | N | N | N |
| Variable | N | N | Y | N | N | Y | N |
| Variables | N | N | N | Y | N | N | N |
| Variant | N | N | N | N | N | Y | N |
| Varying | Y | Y | Y | Y | Y | N | N |
| Vcat | N | N | N | N | N | Y | N |
| Verbose | N | N | N | N | N | N | Y |
| View | Y | Y | Y | Y | Y | Y | N |
| Volatile | N | N | N | N | N | Y | N |
| Volumes | N | N | N | N | N | Y | N |
| Wait | N | N | N | Y | N | N | N |
| Warnings | N | N | N | Y | N | N | N |
| Week | N | N | N | Y | N | N | N |
| When | Y | Y | Y | Y | Y | Y | Y |
| Whenever | Y | Y | Y | N | Y | Y | N |
| Where | Y | Y | Y | Y | Y | Y | Y |
| While | N | N | Y | Y | N | Y | N |
| With | Y | Y | Y | Y | Y | Y | N |
| Without | N | N | Y | N | Y | N | N |
| Wlm | N | N | N | N | N | Y | N |
| Work | Y | Y | Y | Y | Y | N | N |
| Wrapper | N | N | N | Y | N | N | N |
| Write | Y | Y | Y | Y | Y | N | N |
| Writedown | N | N | N | N | Y | N | N |
| Writeup | N | N | N | N | Y | N | N |
| X509 | N | N | N | Y | N | N | N |
| Xa | N | N | N | Y | N | N | N |
| Xid | N | N | N | N | Y | N | N |
| Xml | N | N | N | Y | N | N | N |
| Xmlcast | N | N | Y | N | N | Y | N |
| Xmlexists | N | N | Y | N | N | Y | N |
| Xmlnamespaces | N | N | Y | N | N | Y | N |
| Xor | N | N | N | Y | N | N | N |
| Year | Y | Y | Y | Y | Y | Y | N |
| Year\_Month | N | N | N | Y | N | N | N |
| Years | N | N | N | N | N | Y | N |
| Zerofill | N | N | N | Y | N | N | N |
| Zone | Y | Y | Y | N | Y | Y | N |

## Glossary

| **Term** | **Definition** |
| --- | --- |
| 1NF | First normal form (see the Normalisation appendix) |
| 2NF | Second normal form (see the Normalisation appendix) |
| 3.5NF | See BCNF |
| 3NF | Third normal form (see the Normalisation appendix) |
| 4NF | Fourth normal form (see the Normalisation appendix) |
| 5NF | Fifth normal form (see the Normalisation appendix) |
| 6NF | Sixth normal form (see the Normalisation appendix) |
| Abbreviation | A shortened version of a word or phrase, generally by removing or substituting some characters |
| ABM | Application Business Message, the format of data that is sent from or received by an application that is specific to it |
| Acronym | A word formed from the first or a few key letters in a series of words |
| ALDM | See Application Logical Data Model |
| ANSI | American National Standards Institute, an external body that maintains standards, notably SQL |
| API | Application Programming Interface, an open interface |
| Application Data Model | An Enterprise Architecture model, shows application data entities, used to relate applications to their business concepts |
| Application Logical Data Model | A data model that shows how information in a particular system is structured, independently from its physical implementation |
| Application (Data) Object | An entity that a specific application holds information about. |
| Application Physical Data Model | A data model that describes the physical implementation of data in a specific application |
| Artefact | An enterprise architecture object that represents a physical data object such as a file, field, table or message |
| Association Relationship | An enterprise architecture relationship that shows that two entities are related. Association is the weakest type of relationship, it has no direction and implies no behaviour |
| Attribute | A fact that is known about an entity. An attribute is the logical equivalent of a column |
| Audit Field | A column that is added to a table that captures information about its lineage, such as the date or time it was updated, and users responsible for changes |
| Barker Notation | A data modelling notation |
| Base (Layer) | A data warehouse layer that stores data in a centralised, normalised structure |
| BCNF | Boyce-Codd normal form (see the Normalisation appendix) |
| Boolean | A yes/no field |
| Business Key | A code that is recognised by the business as the identifier of an object |
| Business Object | A thing that the business know facts about |
| Camel Case | Naming an object with the first letter of each word, except the first word in uppercase and the remainder in lowercase. exampleOfCamelCase |
| Candidate Key | A potential combination of attributes used to uniquely identify an entity |
| Cardinality | The quantifier of a relationship between entities, the cardinality expresses how many of entity *x* relate to how many of entity *y* |
| CDM | See Conceptual Data Model |
| Column | A fact held against a row in a table |
| Comment | An annotation that is held against a database object |
| Conceptual Data Model | A high level system agnostic data model that depicts business objects and their relationships |
| Configuration (Layer) | A collection of data warehouse objects that drive the behaviour of processes |
| Constraint | A condition that is evaluated against database actions |
| COTS | Commercial Off The Shelf, a pre-packaged solution that can be bought |
| Data Architecture | A discipline that defines policies, rules and models that manage and describe the enterprise’s information assets |
| Data Warehouse | A collection of information whose primary use is reporting and analytics, where data is sourced from multiple source systems and stored in a non-volatile data structure. |
| Data Warehouse Layer | A logical grouping of data warehouse objects that represents its position in the information pipeline. A data warehouse generally contains landing and staging schemas that handle unprocessed data, a base layer where it is normalised, and a presentation layer where information is optimised for analytics. |
| Datatype | The datatype defines the storage method, acceptable content and format of an attribute. |
| DB2 | A database technology provided by IBM |
| De-normalised | The process of reintroducing data duplication and redundancy to optimise data for end user consumption |
| Definition | The description of an object |
| Degenerate Dimension | An attribute that (after de-normalisation) ceases to be a unique identifier but that still has business value |
| Dimension Table | A de-normalised table, optimised for reporting, and which describes an object that is referenced by a fact. Dimension tables generally contain reference data and master data |
| Dimensional Model | A data model that is optimised for reporting. A dimensional model is generally a de-normalised star-schema or snowflake schema |
| EBM | Enterprise Business Message, a system agnostic message format that is the basis of a Service Oriented Architecture |
| ELDM | See Enterprise Logical Data Model |
| Element | A table, column, entity or attribute |
| Enterprise Architecture | A discipline that describes the components of an organisation, their interactions and goals |
| Enterprise Logical Data Model | Enterprise Logical Data Model, a logical data model that describes the enterprise standard default data structure |
| Entity | In a logical model, an entity is a representation of a table, an object that has properties (attributes) assigned to it |
| External Model | A data model that is provided by a third party |
| Fact Table | In a dimensional model, a fact table holds details about a business event or process |
| Foreign Key | An attribute, or group of attributes that is used to join to another entity |
| Governance | An organisation that manages policy and compliance |
| IDEF1X | Integration Definition for Information Modelling, a modelling standard that includes a data modelling notation |
| Identifying Relationship | A relationship between entities where part of the primary key of an object is the key of another object |
| IE | See Information Engineering |
| Information Engineering | A data modelling notation |
| Initialism | An acronym constructed from the first letters of a sequence of words |
| Intersect Entity | An entity that exists to link two entities |
| ISO | The International Standards Organisation, a body that maintains external standards |
| JSON | JavaScript Object Notation, a markup language used primarily for the exchange of data |
| Junction Entity | See Intersect Entity |
| Landing (Layer) | A data warehouse layer that is used to collect information from source systems |
| Length | In the context of naming standards, the number of characters including spaces.  In the context of attributes, the maximum number of characters that may be stored in a field |
| Logical Data Model | A data model that describes the structure of data independently from its physical implementation |
| Mart (Data) | A repository, downstream of a data warehouse where data is copied and that is used for specific reporting needs |
| Master Data | Data that describes key objects the organisation is comprised of such as customer, employee, locations, and products |
| Metadata | Data that describes data |
| Modifier Term | See Qualifier Term |
| MOTS | Modified Off The Shelf, a technology that is purchased part-built and intended to be a starting point for a development project |
| MSSQL | See SQL Server |
| MySQL | A database technology |
| Naming Standard | A set of rules and patterns that defines how the name of an object should be structured |
| Natural Join | A join that does not specify join criteria, whereby the database automatically infers fields to join on based on common names |
| Natural Key | See business key |
| Non-Identifying Relationship | A relationship to another entity that does not form part of the primary key of the referencing entity |
| Normal Form | A series of rules and standards that describes how well data has been normalised (optimised for the removal of redundancy) |
| Normalisation | The process of increasing the normal form of data by removing redundancy |
| ODBC | Open DataBase Connection, a protocol used to connect to data sources |
| Oracle | In this context, a database technology. Oracle is also the name of a large IT company that owns the database and supplies software, hardware, services and consultancy |
| Pascal Case | Naming an object with the first letter of each word in uppercase and the remainder in lowercase. ExampleOfPascalCase |
| Pattern | A template or standard that is designed for reuse |
| PDM | See Physical Data Model |
| Physical Data Model | A data model that defines the way a data structure has been implemented in a specific storage technology |
| PostgreSQL | A database technology |
| Precision | The number of digits a number may hold |
| Presentation (Layer) | In a data warehouse, a group of objects that are optimised for reporting and user access |
| Primary Key | An attribute or group of attributes that identifies a specific row on a table |
| Prime Business Term | A term that describes an entity in business terms |
| Qualifier Term | A term that refines a prime business term where there are multiple variations of it |
| Realisation Relationship | A relationship that links an abstract version of a concept to a more concrete version of itself. |
| Reference Data | Data that is non-transactional or summary in its nature such as codes and lookup values, configuration data and master data |
| Relational | A database technology or model that stores data in a manner that reduces its redundancy and duplication |
| Relationship | A link between two entities |
| Scale | The number of digits accepted to the right of a decimal point |
| Separator | A character used to separate terms |
| Service Oriented Architecture | An architectural pattern where application components provide services (data exchange and processing) to one another, governed by business contracts |
| Snowflake Schema | A dimensional model that has more than one level of outlying dimension tables around its central fact |
| SOA | See Service Oriented Architecture |
| SQL | Simple Query Language, a language used to access data, usually from relational sources |
| SQL Server | A database technology |
| Staging (Layer) | A data warehouse layer that contains objects that are being processed into the base layer |
| Star Schema | A dimensional model that has only one level of outlying dimension tables around its central fact |
| Structured Data | Data that can be easily interpreted by a machine due to the logical organisation of facts |
| Sub-type | A specialisation of a concept |
| Super-type | A generalisation of a concept |
| Surrogate Key | A meaningless key that is assigned to a piece of data to avoid challenges with business key overlap and change, surrogate keys are usually of data types that are highly performant |
| Systems Integrator | A partner who provides IT implementation services |
| Table | The implementation of a logical entity in a database |
| UML | Unified Markup Language, a modelling notation |
| Unstructured Data | Data that is not easily interpreted by a machine, such as large blocks of text with variable content, images and video |
| Vendor | A party who sells (in this context) IT solutions |
| View | A stored query that is (generally) executed at run time but can be referenced in the same manner as a table |
| XML | eXtensible Markup Language, a markup language usually used in the exchange of data |