Table of Contents

[1 Du site toadword.com-(MySQL) 2](#_Toc436938466)

[1.1 Domains 2](#_Toc436938467)

[Column Domains 2](#_Toc436938468)

[1.2 Naming Foreign Key Constraints 3](#_Toc436938469)

[1.3 Naming Indexes 4](#_Toc436938470)

[1.3.1 Example Index Naming Convention 4](#_Toc436938471)

[1.3.2 Natural or Alternate Keys 4](#_Toc436938472)

[1.4 Primary Keys 5](#_Toc436938473)

[1.4.1 Non-Unique Keys 5](#_Toc436938474)

[1.4.2 Indexes for Foreign Keys 5](#_Toc436938475)

[1.5 Smart Columns 6](#_Toc436938476)

[1.6 Tables 7](#_Toc436938477)

[2 MySQL Coding Standards and Naming Conventions 8](#_Toc436938478)

[3 Oracle – code conventions (Java-pdf) 8](#_Toc436938479)

[4 Code 9](#_Toc436938480)

[5 Android 10](#_Toc436938481)

Naming Conventions

# Du site toadword.com-(MySQL)

<http://www.toadworld.com/platforms/mysql/w/wiki/6103.naming-conventions>

## Domains

<http://www.toadworld.com/platforms/mysql/w/wiki/6106.column-domains>

#### Column Domains

Column naming follows table naming standards very closely except for the addition of domain names at the end of the column name. Column names should

* reflect the data that is stored in the column,
* should, optionally, be suffixed by a domain
* should not be overloaded (used for different data elements depending on the circumstances)
* should not contain abbreviations unless necessary.
* should not contain smart codes or columns

It is possible to include a comment for each column in the definition of each table. For a large database model this can be a very large overhead to create and maintain. However, if columns are named meaningfully and the column type can be deduced from the column name, then there is no need to maintain a data dictionary. Strong naming conventions in the database also lead to more readable code in the application layer.

If you choose to implement a domain system the following guidelines are very useful.

All columns should contain a domain representation that specifies the type of data that the column contains. This domain representation is overloaded somewhat because it specifies not only the database type and length, but also what kind of data is being stored and its general use in the application. For example, the \_SN domain could represent a MySQL data type of VARCHAR with a length of 50, but also tells us that the data stored in the column is short name kind of data.

If none of these domain specifications fit for the type or kind of data that needs to be captured, the application developers and DBA need to establish a new domain name to fit the data. This also gives time to review the data to make sure that it is being stored properly.

***Table 1: Example Domains***

|  |  |  |  |
| --- | --- | --- | --- |
| Domain | MySQL Type | Description | Example |
| \_ID | unsigned int | primary or foreign key | student\_id |
| \_AMT | decimal(7,2) | dollar amounts for money | outstanding\_amt |
| \_DT | date | date with no time | birth\_dt |
| \_DTM | datetime | date with time | submission\_dtm |
| \_TM | datetime | Time only (Date is always 12/31/1899) | start\_tm |
| \_SN | varchar(20) | short name (typically a user abbreviation of the long name) | company\_sn |
| \_LN | varchar(50) | long name (full name of something) | company\_ln |
| \_TXT | text | used for long text | curriculum\_txt |
| \_IND | enum('Y','N') | indicator | active\_ind |

Implementation of a domain system is a difficult up-front task. However, it pays large dividends during development and maintenance of the application. Consider the use of domains for the following:

* An entity relationship diagram serves the purpose of the data dictionary as well.
* Automatically generate validation code at the application and browser level for forms.
* Each domain needs only one validation function.

## Naming Foreign Key Constraints

<http://www.toadworld.com/platforms/mysql/w/wiki/6109.naming-foreign-key-constraints>

Foreign key constraint names should always begin with FK\_ followed by the parent table name, followed by a dollar sign, followed by the name of the child table. As with indexes, if more than one relation exists between the same two tables, the name of the constraint will be followed by a number.

fk\_teacher$student

This naming scheme can produce constraint names that exceed MySQL's size, so abbreviations are acceptable. Try to be consistent with the abbreviation of a table name. For instance, if you abbreviate STUDENT as STU in a constraint name try to do this each time it is necessary to use an abbreviation for that table. Look for examples in the existing model before using your own.

## Naming Indexes

<http://www.toadworld.com/platforms/mysql/w/wiki/6108.naming-indexes#example-index-naming-convention>

* [Example Index Naming Convention](http://www.toadworld.com/platforms/mysql/w/wiki/6108.naming-indexes#example-index-naming-convention)
* [Natural or Alternate Keys](http://www.toadworld.com/platforms/mysql/w/wiki/6108.naming-indexes#natural-or-alternate-keys)
* [Non-Unique Keys](http://www.toadworld.com/platforms/mysql/w/wiki/6108.naming-indexes#non__002dunique-keys)
* [Indexes for Foreign Keys](http://www.toadworld.com/platforms/mysql/w/wiki/6108.naming-indexes#indexes-for-foreign-keys)

MySQL does not require the naming of indexes or constraints. When such an object is created, it is assigned a default system name. Unfortunately, system specified names are not usually useful to the human user for tuning and debugging.

### Example Index Naming Convention

MySQL has two basic index types - unique and non-unique. In addition, sometimes an index is put in place to support a foreign key constraint. Consistent naming conventions make the creation of large migration scripts useful because it is easy to locate similar objects by commonalities in their names.

### Natural or Alternate Keys

In a model that uses surrogate keys unique indexes are often enforced on natural keys. A basic example would be in a table of current United States citizens, the social security number (SSN) should be unique. Data modelers will often not use this as a primary key however because there are rare instances when a person SSN can change (e.g. identity theft) or when an SSN is duplicated (e.g. administrative error, or recycling old numbers). Instead, the modeler will use a surrogate key that has no meaning and enforce a unique index on SSN and date of birth (DOB). Since SSN and DOB are attributes of a person that occur naturally, this is known as a natural or alternate key.

Alternate keys in this convention will be named by prefixing AK\_ to the table name. If there is more than one natural key in an entity, the name will be followed by a number. For example in the table PERSON, the alternate keys would be named:

* ak\_person
* ak\_person2

Abbreviations of the table name are unacceptable unless the index name is longer than the server allows. In this case a sensible abbreviation should be used.

Notice that the column(s) of the index do not play a part in its name. Such a name would be quite long, especially for a composite key. If more information about an index is necessary, the table definition can be consulted with the SHOW CREATE TABLE PERSON statement.

## Primary Keys

<http://www.toadworld.com/platforms/mysql/w/wiki/6105.primary-keys>

Every table should have a primary key. When using surrogate keys (a good idea), they should all be of exactly the same type. A good type to use is unsigned integer because it allows for over 4 billion rows before a migration must occur.

The name of the primary key column should reflect the table name and the domain \_ID. This provides an effective way to keep the primary key of each table unique and meaningless. Using this method instead of allowing the primary key to be data allows the users to change the data without needing to cascade data changes to all the affected tables. As an example, the primary key for the STUDENT table is student\_id.

The primary key constraint and index names in MySQL are system generated and cannot be overridden. The primary key of each table in MySQL has the name PRIMARY.

### Non-Unique Keys

Typically non-unique keys are used for performance reasons. If a column often appears in a WHERE clause it generally needs an index or to be part of a composite index for performance reasons.

Non-unique keys in this convention are named by prefixing IX\_ to the table name. If there is more than one non-unique key in an entity, the name will be followed by a number. For example, in the table PERSON, the alternate keys would be named:

* ix\_person
* ix\_person2

Abbreviations of the table name are unacceptable unless the index name is longer than the server allows. In this case a sensible abbreviation should be used.

### Indexes for Foreign Keys

In highly normalized models employing foreign key constraints, it is useful to get a message back that names the problem. By providing special naming to the indexes that support foreign keys, it is possible to trap and handle errors involving referential integrity at the application level.

Since an index supporting a foreign key can be either unique or non-unique it makes sense to name simply append FK\_ to the index name. For unique indexes that support foreign keys on a table call PERSON the names look like

* ak\_fk\_teacher$student
* fk\_ak\_teacher$student

For a non-unique index that supports foreign keys on a table named PERSON the names look like

* ix\_fk\_teacher$student
* fk\_ix\_teacher$student

Whether the FK\_ is first or second in the name, consistency is the key so that users of the database can depend on the names to give them the information they need.

## Smart Columns

<http://www.toadworld.com/platforms/mysql/w/wiki/6107.smart-columns>

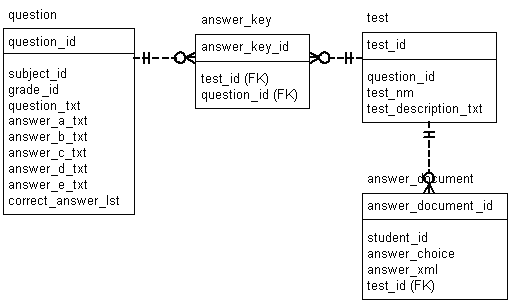
In the state of Texas a school is identified by its nine digit county district code number (CDCN). The first six digits of the number represent the district's code and the last three digits represent the campus number. Furthermore, in the last three digits, numbers lower than ten are high schools, numbers above 40 are intermediate schools and numbers above 100 are elementary schools.

So, for Washington High School in the Ranch School district the CDCN is 101902004. It would seem that this number should go into a column nicely, however this is not the case. Any time it would be necessary to look up the building by its building number or too look up the district by its number this column would have to be parsed by the database. Any inconsistency in the number could wreak havoc on an application. A strong case can be made for splitting such a "smart code" into two columns that represent the district\_number and the building\_number.

The above example is common in dealing with governmental organizations. Each case should be considered individually when determining how to store such information, but you should always be wary of such "smart codes." Laws and policies have a way of changing over time.

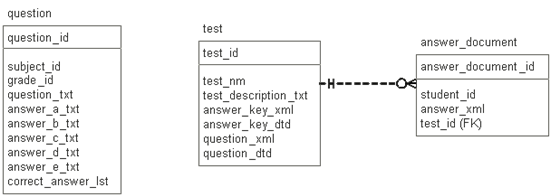
A more dangerous use of "smart columns" comes from application developers who would wish to solve a short term problem. If, in the case of the CDCN above, a district gets large enough to have more than 10 high schools, the definition would have to change and a migration would need to take place. Such short term convenience is very often outweighed by harsh long term consequences.

A common exception to this thinking can be demonstrated in considering the idea of storing test keys in a system where thousands of tests made up of hundreds of questions are written to the database every marking period in a school year.



The model above is normalized and provides referential integrity. The problem is that as the tests come in, the associative table grows very quickly. The above model also demonstrates another exception. By the coding standards described above the table ANSWER\_KEY should be named TEST\_QUESTION, however the common name for the table that represents this functionality is so frequently used by the business that the business name was used instead.

The way to eliminate this problem is by storing the entire test key in a single column. A good technology for this is XML because the DTD or schema can be stored with it for the purpose of validation that would normally be performed by the database.



In the diagram above there is no constraint between the questions and the test. The application must take care of this. In this case the DTD from the test will insure the answer document is correct. This is the trade off for speed.

Comparisons can now be made between the answer\_key\_xml column and the answer\_xml column to score an exam while validation of the answer\_key, questions and answers can be taken care of by the DTDs stored in the TEST table.

## Tables

Table names should reflect the use of the table with no abbreviations unless necessary to fit the name size limitations of MySQL. When multiple words are needed to specify the table name, the words should be separated by underscores.

It is very important to understand that when MySQL is implemented on an operating system that is case sensitive, its table names are case sensitive unless the lower\_case\_table\_names variable is set in the my.cnf file. For a database using the InnoDB storage engine to be portable across all operating systems, it is necessary for all the tables to have lower case names. So, all table names should be lower case.

Here are some examples of appropriate table names:

* student
* master\_schedule
* security\_user\_persmission

If a table is used to create a many-to-many relationship between two tables, both source table names should be used in the associative table name. For example:

* student\_homework
* lesson\_resource

Again, abbreviated names should be avoided unless necessary to fulfill MySQL table naming size limits.

Tables that are part of a specific subsystem of an application may have an abbreviation of the sub system prefixed to the table name. For example, tables the audit subsystem might be prefixed with AUD\_ (ex: aud\_row or aud\_column).

Some examples of inappropriate table names are:

* stu - (unnecessary abbreviation)
* ms\_sched - (unnecessary abbreviation)
* MASTER\_SCHEDULE - (upper case)
* MasterSchedue - (camel case and no underscore between words)

# MySQL Coding Standards and Naming Conventions

<http://www.realdealmarketing.net/docs/mysql-coding-standards.php>

<http://anandarajpandey.com/2015/05/10/mysql-naming-coding-conventions-tips-on-mysql-database/>

# Oracle – code conventions (Java-pdf)

<http://www.oracle.com/technetwork/java/codeconventions-150003.pdf>

Les points:

-Éviter les fichiers plus longs que 2000 lignes

-La classe ‘public’ devrait être la première

-Avant chaque classe, fonction, fournir la date, copyright notice, brève description du programme. 'bloc comment

/\*

\* Classname

\*

\* Version info

\*

\* Copyright notice

\*/

-Indentation (4 espaces), tab sont de 8 espaces

-Max 70 caractères sur une ligne

-Mette plusieurs sur plusieurs lignes qui en avait juste une, couper après une virgule, un opérateur.

* -Éviter de dupliquer les commentaire
* Bloc/single line comment, trailing comment /\* \*/, end of line comment // pour mettre du code en commentaire pas des commentaires avec //

-Documentation (Documentation comments – fichier HTML)

-CODE

1. une déclaration de variable ou constante par ligne (section 6 : Declaration)
2. Section 7 : Statements (les mots réservés en MAJUSCULE)
3. Section 8 : White Space
4. Section 9 : Naming Conventions
5. Section 10 : Programming Practices

# Code

<http://www.williamrobertson.net/documents/plsqlcodingstandards.html>

-Documenter en français

-Les mots réserver MAJUSCULE, le reste en minuscule

-Indentation

-dans les commentaires éviter de faire du fla fla, facile à faire et à maintenir

-Éviter de recycler les variables

-Enlever les variables non utilisée des programmes

-En PL/SQL utiliser %TYPE

-Ne pas utiliser PL/SQL auqn je peux utiliser des commandes SQL

-Si des constantes changent en variable (avoir un nom différent pour chaque)

-Dans le SQL (ALIGNER à gauche au lieu d'aligner à droite:

update employees

set hire\_date = sysdate

where hire\_date is null

and termination\_date is null;

Instead, use:

* select last\_name, first\_name

from employees

where department\_id = 15

and hire\_date < sysdate;

-Dans le Where clause, mettre une opération par ligne

-Ne pas trop ajouter de ligne vide dans le SQL

-Ajouter des abbréviations signifiantes comme alias dans les SQL

# Android

<http://source.android.com/source/code-style.html>

Sommaire:

* Les exceptions
* 'Finalizers'
* Ne pas utiliser**, import foo.**\*, les compléter
* Pour le nouveau code, ne jamais utiliser les librairies dépassées (depreciated)
* Javadoc (Copyright, décrire les classes et interfaces)
* Écrire des méthodes courtes
* Limiter l'utilisation des variables locales et déclarer là où elles sont utilisées
* L'ordre des Import: 1) Android, 2) troisième partie, 3) java et javax
* Convention des noms :
  + Non-public, non-static field names start with m.
  + Static field names start with s.
  + Other fields start with a lower case letter.
  + Public static final fields (constants) are ALL\_CAPS\_WITH\_UNDERSCORES.
  + Limiter les longueurs de caractères (100), dans un autre site 70 max
  + Traiter les acronymes comme des noms ex : class Html, non : class HTML
  + Utiliser les TODO comments
  + Utiliser le Log très peu souvent
  + Être consistent