# Coding Guidelines | Database Standards

**Normalization**

Normalization must be used as required, to optimize the performance. Under-normalization will cause excessive repetition of data, over-normalization will cause excessive joins across too many tables. Both of them will get worse performance.

Use denormalization to avoid joins and improve performance. e.g. Lets assume a table for doctors with their specialty. One way is to have a specialty table and keep the specialty\_id in doctors table as a foreign key. We can put the specialty\_name also in the doctor table. This will reduce joins and improve performance.

**Naming**

Use meaningful names for tables and columns.

Follow a consistent naming convention throughout. If you are using snake case for names (e.g. purchase\_detail) use it for all tables. Do not mix different cases. The following naming convention is fixed for our projects...

Tables: lower snake case (lowercase with underscores). e.g. purchase\_detail

Fields: lower snake case (lowercase with underscores). e.g. is\_approved, deleted\_at etc. Index: lower snake case (lowercase with underscores) and idx\_ prefixed. e.g. idx\_email

Keys: lower snake case (lowercase with underscores) and fk\_ or pk\_ prefixed. e.g. pk\_user\_id, fk\_user\_id

Use singular for table names (i.e. use student instead of students). Table represents a collection of entities, there is no need for plural names.

Don’t use spaces for table and column names. Otherwise you will have to use ‘{‘, ‘[‘, ‘“’ etc. characters to access them. Don’t use unnecessary prefixes or suffixes for table names (i.e. use School instead of TblSchool, SchoolTable etc.).

**Constraints**

Always have a PrimaryKey (preferably integer). Generate it if there isn't one. Define all foreign keys properly for data integrity.

Clearly specify nullability for all fields.

Use default and check constraints where required.

**Datatypes and Characterset**

Use integer datatype for primary keys and foreign keys.

Use bit or boolean datatype for boolean fields. Using integer or varchar is not efficient. Also start those column names with “Is”. UTF8 characterset is good in most of the cases. But to store extended characters and emoticons use UTF8MB4 characterset. Keep passwords encrypted. One way encryption is better.

**Index**

All primary keys will automatically be indexed. Implement index on all foreign keys.

Implement index wisely on all keys that are being used in where clauses a lot. Don't overuse index. It will slowdown DML operations.

**Stored Procedures**

Create stored procedures if you want to use multiple tables and do some small processing. Don't write business logic inside stored procedures. Perform only database operations.

**Security and Documentation**

Don't use root user for database access in code.

Document database design with ER diagram. Also write comment lines for your triggers, stored procedures and other scripts. Database server and the web server should be placed in different machines. This will provide more security (attackers can’t access data directly) and server CPU and memory performance will be better because of reduced request number and process usage.

**Other**

Don't store images in database. Store image files in a directory and keep their names in database. Avoid “select \*” queries. Use "select <required\_columns\_list>" for better performance.

Measures query performances by explaining its plan.

Have archive policy on database. This helps us to manage database size and performance.