SQL Do's and Don'ts

10 Common *traps and tricks* to help you become a magical query fairy

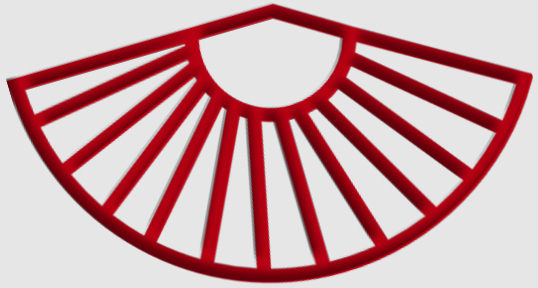
Danielle Lynch
Level Data Analytics
Northeastern University
March 2019

No | Negative Searches

Try to avoid the `NOT` or <> symbol operator. SQL will search every single row to identify that they ALL belong or exist within the table.

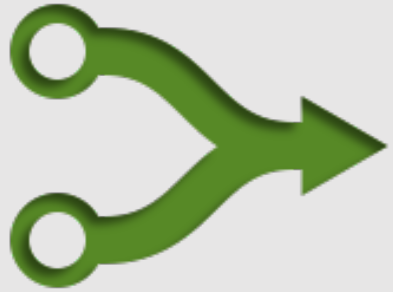
Yes | Positive Operators

It is much faster to search for an exact match using the `LIKE`, `IN`, `EXIST` or = symbol operator.



No | Fanouts

Certain aggregations can misbehave if used against joined tables, creating ‘fanouts’ where the joined table has more rows than the primary table.



Yes | Safe Joins

Count rows before/after the `JOIN`. If needed, begin 1-to-many joins with the most granular table or group data for a 1-to-1 relationship.

No | Redundancy

Data redundancy is good for backups but not for table data. Don’t repeat data in other locations.

Yes | One Location

Always keep data in one location and use relationships between primary and foreign keys to query the data.

No | Queries gone wild

Avoid `DISTINCT`, `SELECT` *, and leading wildcards in `WHERE LIKE` statements, which incur extra operations and slow queries down.

Yes | Be specific

Define each column to limit the size of the record set. Only use leading wild cards when absolutely necessary.

No | Inconsistency

There are many styles for naming primary and foreign keys. Look out for foreign keys that don't obviously match up to a table.

Yes | Naming Conventions

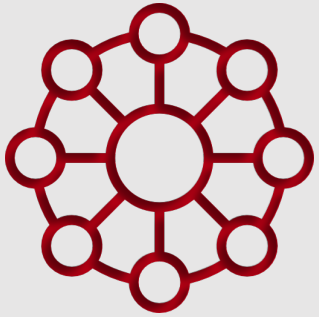
The most popular naming method is to have a primary key `id` for any table `foo`, and that all foreign keys be named `foo_id`.

No | Double Dipping

‘Double dipping’ is accessing the same data twice. Don’t do it! Try to get all the jobs done in a single query visit whenever possible.

Yes | Temp Tables

Use temporary tables wisely - query only the large tables once. This reduces the power required in processing.



No | Too Many Joins

The cost of planning and optimizing a query with too many joins can become problematic.

Yes | Smaller is Better

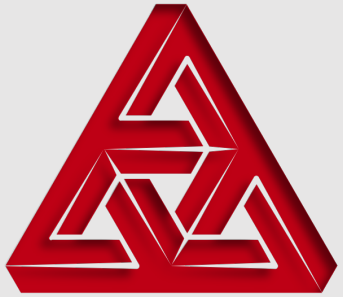
It’s better to have a dozen or fewer tables per query if you need queries to execute very fast with high concurrency.

No | Mixed Data Types

Mixing different data types can not only cause performance problems but implicit type conversions can create hard-to-find errors.

Yes | Matching Identifiers

Try to use the same data types to store similar or related values, especially if they'll be used in a `JOIN` condition.

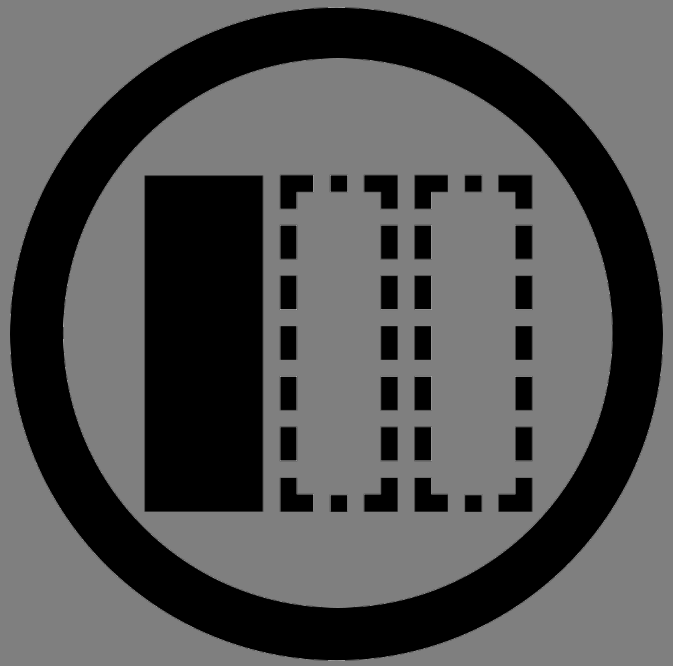


No | Excessive Nesting

Avoid views that call views that call views … and subqueries that rely on outer queries, which painstakingly compare values row by row.

Yes | Less is More

Stick to necessary joins and columns and smaller data types. `LIMIT` can also save valuable computation time.



At the Beginning, Consider the End

Remember that NULL values aren't the same as, for instance, a zero-length string. NULL is neither a character nor number. It is an empty variable. As such, comparing NULL with anything will always return NULL. It is often recommended to avoid NULL unless it's actually the right way to model your data's reality.

The NULL vs. empty string debate between database administrators has been ongoing for decades. You can choose to use NULL values when no value is present or you can use actual literal values such as zero-length strings or 0 integer values. ***What you use in the database should be uniform across all tables or queries can get messy.***