

数据库表名命名规范：如何避免混乱与错误

在实际开发中，一个看似微小的表名命名问题，往往会在团队协作、系统维护或跨项目对接时引发连锁反应。尤其在使用MySQL、PostgreSQL等主流数据库时，表名不规范不仅影响代码可读性，还可能因大小写敏感、特殊字符冲突等问题导致运行时异常。因此，建立一套清晰、统一的数据库表名命名规范，是每个开发者必须掌握的基础能力。

表名应以英文小写为主，避免大小写混用

多数数据库系统对表名的处理方式不同，例如MySQL在Windows下不区分大小写，但在Linux环境下则严格区分。这种差异容易在部署迁移时埋下隐患。因此，最稳妥的做法是始终使用小写字母命名表名，杜绝大写或驼峰式命名（如`UserProfile`）。即使在某些工具中支持大写，也建议统一为小写，确保跨平台一致性。此外，避免使用保留字如`select`、`order`、`group`等作为表名，即便加引号包裹也可能带来不必要的复杂性。

使用下划线分隔单词，提升可读性

当表名由多个单词组成时，采用下划线（snake_case）是最广泛接受的方式。例如`user\_profile`、`order\_item`、`product\_category`，相比驼峰命名（`UserProfile`），更符合数据库命名惯例，且在查询语句中更易阅读。虽然部分团队偏好驼峰命名以契合编程语言风格，但数据库本身并不具备“驼峰”概念，强行使用反而增加歧义。尤其在编写SQL脚本时，`SELECT \* FROM user\_profile`比`SELECT \* FROM UserProfile`更直观，不易出错。

避免使用缩写和模糊词汇

常见误区是过度使用缩写，如`usr`代替`user`、`ord`代替`order`，或用`tbl\_`前缀标记表类型。这类做法短期内节省输入时间，长期来看却严重损害可读性。例如`tbl\_usr`难以让新成员快速理解其含义，而`user`或`user\_account`则一目了然。同样，避免使用`info`、`data`、`temp`等泛化词汇，它们无法准确表达表的实际用途。若需区分表类型，可通过业务领域命名体现，如`user\_login\_log`、`payment\_transaction`。

命名应反映业务实体，而非技术结构

表名应体现其承载的数据主体，而不是存储方式或技术实现。例如，不应将日志表命名为`log\_table`，而应具体为`access\_log`、`error\_log`；订单相关表不应叫`order\_data`，而应为`order\_header`、`order\_detail`。这种命名方式有助于在设计阶段就明确数据模型边界，也便于后续的权限控制、索引优化和数据归档策略制定。同时，避免在表名中加入版本号或时间戳（如`user\_v2`、`order\_2024`），这些信息应通过其他机制管理，而非固化在名称中。

建议命名	不推荐命名	说明
------	-------	----

-----	-----	-----
-------	-------	-------

`user_profile`	`usr_profile`	缩写降低可读性
----------------	---------------	---------

`order_item`	`order_item_temp`	临时表应通过其他方式标识
--------------	-------------------	--------------

| `payment\_record` | `pay\_rec` | 过度缩写易混淆 |

| `audit\_log` | `log\_audit` | 语序不当，逻辑不清 |

特殊字符与长度限制需注意

尽管大多数数据库允许使用下划线、数字和字母组合，但禁止使用空格、括号、点号、斜杠等符号。部分系统对表名长度有限制（如MySQL默认64字符），因此应尽量简洁。一个合理的命名长度通常在15~30个字符之间，既能表达完整语义，又不会过长。如果遇到复杂场景，可通过合理拆分表结构来解决，而非堆砌冗长名称。

常见问题

Q：是否需要在表名前加前缀？

A：除非有明确的多租户或模块隔离需求，否则不建议添加`app\_`、`sys\_`等通用前缀，容易造成命名冗余。

Q：是否可以使用中文命名？

A：理论上可行，但会带来编码、工具兼容性和维护成本问题，强烈不推荐。