阿里面试：MySQL 一个表最多 加几个索引？ 6个？64个？还是多少？

原创 尼恩架构团队 技术自由圈 2025年05月26日 09:41 湖北

FSAC未来超级架构师

架构师总动员
实现架构转型，再无中年危机

技术自由圈
疯狂创客圈（技术自由架构圈）：一个 技术狂人、技术大神、高性能 发烧友 圈子。圈内一...
293篇原创内容

公众号

尼恩说在前面：

在40岁老架构师 尼恩的**读者交流群**(50+)中，最近有小伙伴拿到了一线互联网企业如得物、阿里、滴滴、极兔、有赞、shein 希音、shopee、百度、网易的面试资格，遇到很多很重要的面试题：

> - mysql中，一个表最多只能加多少个索引嘛？一个联合索引最多只能多少列呢？
> - 索引加多了，会存在哪些问题呢？
> - InnoDB存储引擎
> - MyISAM存储引擎
> - 一个表设计多少个索引合理呢？
> - 索引设计过多存在哪些问题？
> - 阿里巴巴编程规范中， 单表索引数量，建议控制在5个以内 ，为什么？

前几天 小伙伴面试阿里，遇到了这个问题。但是由于 没有回答好，导致面试挂了。

小伙伴面试完了之后，来求助尼恩。那么，遇到 这个问题，该如何才能回答得很漂亮，才能 让面试官刮目相看、口水直流。

所以，尼恩给大家做一下系统化、体系化的梳理，使得大家内力猛增，可以充分展示一下大家雄厚的 “技术肌肉”，**让面试官爱到 “不能自已、口水直流”**，然后实现”offer直提”。

当然，这道面试题，以及参考答案，也会收入咱们的 《尼恩Java面试宝典》V145版本PDF集群，供后面的小伙伴参考，提升大家的 3高 架构、设计、开发水平。

> 最新《尼恩 架构笔记》《尼恩高并发三部曲》《尼恩Java面试宝典》的PDF，请关注本公众号【技术自由圈】获取，后台回复：领电子书

1. 索引基础 ：一个 表的 "数据目录"

1.1 什么是 索引？

- 没有索引：数据库要扫描整张表，就像你从图书馆第一本书开始找
- 有索引：直接定位到数据位置，效率提升几十甚至上百倍

索引就是"数据的目录" ， 想象一下你去图书馆找书，没有目录的话你得一本本翻，有了目录就能直接找到想要的书。

索引就是数据库的"数据目录"，它能帮你快速定位数据。

在MySQL中，索引 是一种特殊的数据结构，通常是B+树，它存储着字段值 和对应记录的主键值。

1.2 为什么需要索引？

没有索引时，数据库要执行全表扫描，就像你从图书馆第一本书开始一本本找，数据量越大查询越慢。

有索引后，数据库可以直接定位数据位置，效率提升几十甚至上百倍。

数据量越大， 索引的价值就大， 在百万级、甚至千万级的 表中，有索引的查询可能只需几毫秒，没索引可能要几秒， 甚至更久。

但凡事都有不利的一面， 索引不是万能的，它是以额外存储空间和写入性能为代价换取查询速度的提升，需要权衡利弊。

1.2 索引的常见类型

索引的常见类型 有：

- 普通索引：最基本的索引，没有任何限制
- 唯一索引：要求索引列的值必须唯一

- 主键索引：特殊的唯一索引，不允许有空值
- 联合索引：多个列组合的索引

普通索引是最基本的索引类型，没有任何限制，允许重复值和空值。

唯一索引要求索引列的值必须唯一，但允许有空值。

主键索引是特殊的唯一索引，不允许有空值，每个表只能有一个。

联合索引是多个列组合的索引，遵循最左前缀原则。

此外还有全文索引、空间索引等特殊类型。

不同类型的索引适用于不同场景，比如用户名适合用唯一索引，文章内容适合用全文索引。

2. InnoDB存储引擎 的 索引 限制

InnoDB存储引擎 是 MySQL 最常用的存储引擎，

InnoDB 作为MySQL5.5后的默认引擎，InnoDB支持事务、行级锁、外键约束等高级功能。

InnoDB 的索引采用聚簇索引（主键索引） + 非聚簇索引 （二级索引） 结合的结构，主键索引的叶子节点直接存储行数据，这使得主键查询特别高效。

InnoDB还支持MVCC多版本并发控制，大大提高了并发读写性能。

对于大多数业务场景，InnoDB都是最佳选择。

InnoDB是MySQL最常用的存储引擎，它就像一辆高性能跑车，既稳定又快速。

InnoDB存储引擎 索引数量限制

- 最多64个普通索引 + 1个主键索引 = 65个
- 每个索引最多包含16个字段

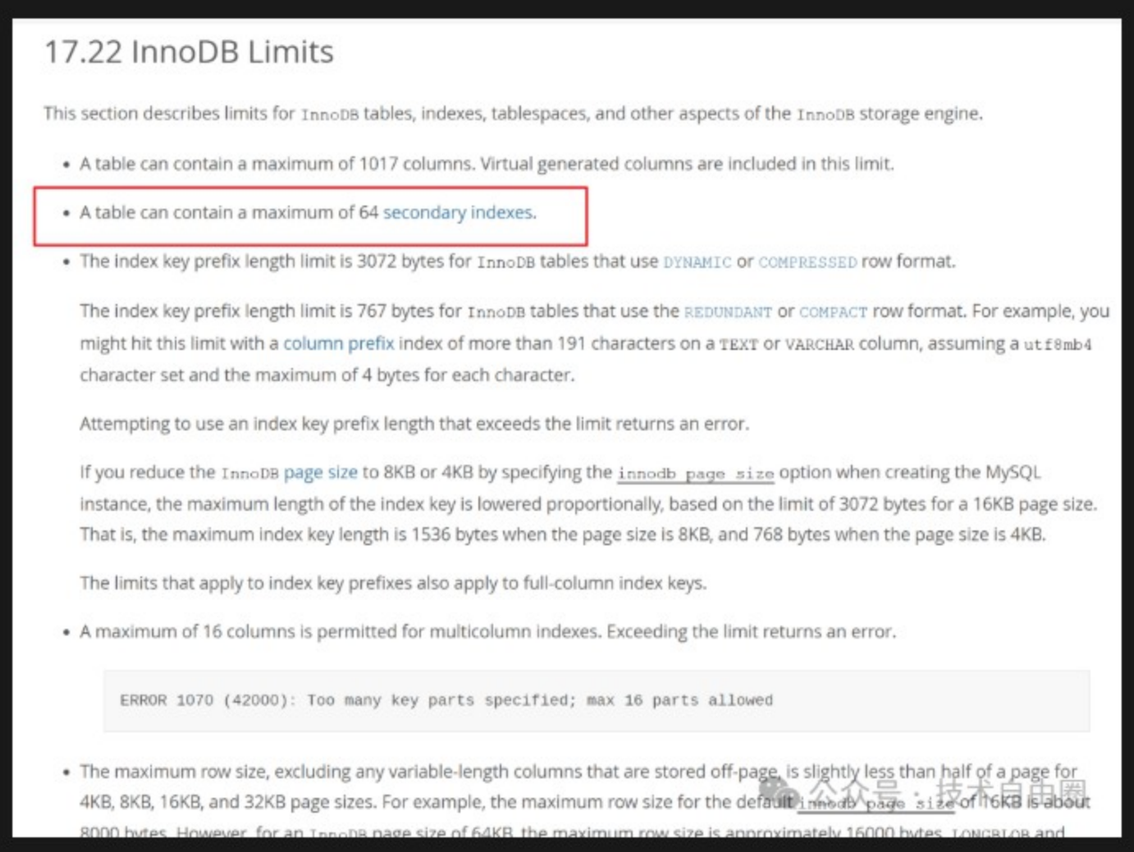
尼恩对索引 使用建议：

- 虽然能创建 65个，除非迫不得已，不建议这么干！
- 就比如说，像一个人能吃10碗饭，不代表就一定要吃10碗。

2.1 InnoDB 索引 的 数量限制

根据MySQL官方文档， InnoDB存储引擎 它最多允许 一个表最多 64个二级索引（即非主键索引），

官方文档有说明 如下



17.22 InnoDB Limits

This section describes limits for InnoDB tables, indexes, tablespaces, and other aspects of the InnoDB storage engine.

- A table can contain a maximum of 1017 columns. Virtual generated columns are included in this limit.
- A table can contain a maximum of 64 secondary indexes.
- The index key prefix length limit is 3072 bytes for InnoDB tables that use DYNAMIC or COMPRESSED row format.

  The index key prefix length limit is 767 bytes for InnoDB tables that use the REDUNDANT or COMPACT row format. For example, you might hit this limit with a column prefix index of more than 191 characters on a TEXT or VARCHAR column, assuming a utf8mb4 character set and the maximum of 4 bytes for each character.

  Attempting to use an index key prefix length that exceeds the limit returns an error.

  If you reduce the InnoDB page size to 8KB or 4KB by specifying the innodb_page_size option when creating the MySQL instance, the maximum length of the index key is lowered proportionally, based on the limit of 3072 bytes for a 16KB page size. That is, the maximum index key length is 1536 bytes when the page size is 8KB, and 768 bytes when the page size is 4KB.

  The limits that apply to index key prefixes also apply to full-column index keys.
- A maximum of 16 columns is permitted for multicolumn indexes. Exceeding the limit returns an error.

```
ERROR 1070 (42000): Too many key parts specified; max 16 parts allowed
```

- The maximum row size, excluding any variable-length columns that are stored off-page, is slightly less than half of a page for 4KB, 8KB, 16KB, and 32KB page sizes. For example, the maximum row size for the default innodb_page_size of 16KB is about 8000 bytes. However, for an InnoDB page size of 64KB, the maximum row size is approximately 16000 bytes. LONGBLOB and

公众号：技术自由圈

链接如下：

> dev.mysql.com/doc/refman/...

InnoDB最多允许64个二级索引（非主键索引），当然， 还有 加上1个主键索引，总共65个索引。

那在InnoDB中，一个表，最多可以有 64+1=**65** 个索引

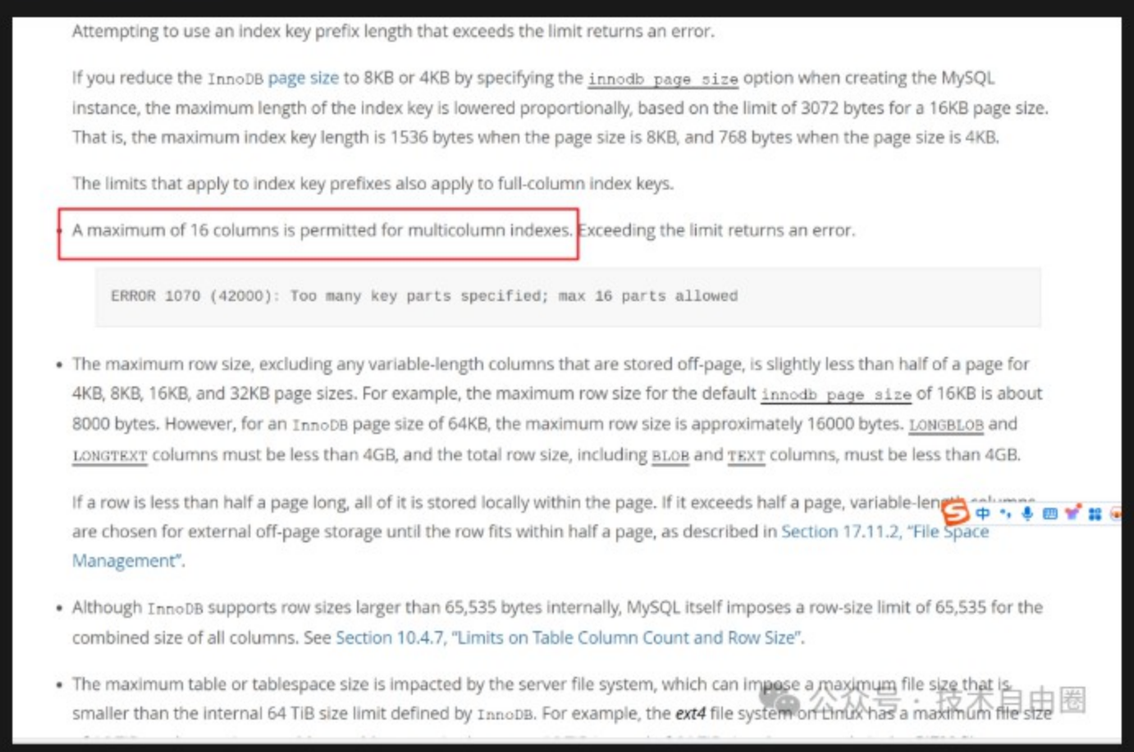
而对于一个索引，最多有多少列呢？

2.2 InnoDB 索引列 的 数量限制

InnoDB 中，一个 索引 最多 能允许 多少个 列 ？

结论是： 一个 索引 最多是16列。

官方文档也是有说明的：



Attempting to use an index key prefix length that exceeds the limit returns an error.

If you reduce the InnoDB page size to 8KB or 4KB by specifying the innodb_page_size option when creating the MySQL instance, the maximum length of the index key is lowered proportionally, based on the limit of 3072 bytes for a 16KB page size. That is, the maximum index key length is 1536 bytes when the page size is 8KB, and 768 bytes when the page size is 4KB.

The limits that apply to index key prefixes also apply to full-column index keys.

- A maximum of 16 columns is permitted for multicolumn indexes. Exceeding the limit returns an error.

```
ERROR 1070 (42000): Too many key parts specified; max 16 parts allowed
```

- The maximum row size, excluding any variable-length columns that are stored off-page, is slightly less than half of a page for 4KB, 8KB, 16KB, and 32KB page sizes. For example, the maximum row size for the default innodb_page_size of 16KB is about 8000 bytes. However, for an InnoDB page size of 64KB, the maximum row size is approximately 16000 bytes. LONGBLOB and LONGTEXT columns must be less than 4GB, and the total row size, including BLOB and TEXT columns, must be less than 4GB.

  If a row is less than half a page long, all of it is stored locally within the page. If it exceeds half a page, variable-length columns are chosen for external off-page storage until the row fits within half a page, as described in Section 17.11.2, "File Space Management".

- Although InnoDB supports row sizes larger than 65,535 bytes internally, MySQL itself imposes a row-size limit of 65,535 for the combined size of all columns. See Section 10.4.7, "Limits on Table Column Count and Row Size".

- The maximum table or tablespace size is impacted by the server file system, which can impose a maximum file size that is smaller than the internal 64 TiB size limit defined by InnoDB. For example, the *ext4* file system on Linux has a maximum file size

公众号 · 技术自由圈

链接如下：

> dev.mysql.com/doc/refman/...

每个索引最多可以包含16个字段，这意味， 可以创建一个包含16个字段的超级联合索引。

但要注意，这些是理论最大值，实际应用中应该远低于这个限制。

索引越多，或者一个索引里边的 列越多， 维护成本越高，特别是对于写入频繁的表，过多的索引会严重影响性能。

通常建议单表索引不超过5-8个，核心查询字段优先建索引。

对于联合索引，字段数最好控制在3-5个以内。

尼恩 建议是 ： 定期使用EXPLAIN分析查询语句，确保索引被正确使用，删除冗余和低效的索引。

3. MyISAM存储引擎 的 索引 限制

MyISAM 是 MySQL早期的默认存储引擎，虽然现在用得少了，但在某些场景下仍有价值。

MyISAM 适合读多写少的场景。

MyISAM 不支持事务和行级锁，但查询速度非常快，特别适合读多写少的场景。

> 尼恩提示：MyISAM的表级锁在写入时会锁定整个表，不适合高并发写入场景。

3.1 索引数量限制

MyISAM每个表最多支持64个索引，主键索引不计入此限制。

每个索引最多可以包含16个字段，与InnoDB相同。

MyISAM的索引使用B-tree结构存储，支持前缀索引，可以只对字段的前N个字符建立索引。

MyISAM存储引擎 的 索引 限制 如下：

- 最多64个索引（主键不算在内）
- 每个索引最多16个字段

3.2 MyISAM 与InnoDB的区别

MyISAM和InnoDB的主要区别包括：

- MyISAM 不支持事务
- MyISAM 表级锁（不是行级锁）
- MyISAM 适合读多写少的场景

MyISAM不支持事务，而InnoDB支持；

MyISAM不支持事务，而InnoDB支持；

MyISAM只有表级锁，InnoDB支持行级锁；

MyISAM不支持外键，InnoDB支持；

MyISAM的崩溃恢复能力较弱，InnoDB更可靠；

MyISAM的全文索引较早出现，但现在InnoDB也支持了。

选择存储引擎时，如果不需要事务且读多写少，可以考虑MyISAM，否则应该选择InnoDB。

4. 索引数量：少即是多

在数据库设计中，索引数量应该遵循"少即是多"的原则。

过多的索引不仅不能提高性能，反而会带来各种问题。

索引就像书中的目录，一本几百页的书有3-5个目录章节就足够了，如果每页都做一个目录，反而会让查找变得困难。

Captured by FireShot Pro: 12 8月 2025, 08:31:20
https://getfireshot.com

数据库索引也是如此，需要精心设计，只给真正需要的查询条件建立索引。

4.1 阿里巴巴规范建议

日常开发中，一个表设计多少个索引合适呢？

> 阿里巴巴《Java开发手册》技术文档，**单表索引数量建议控制在5个以内, 单个索引的字段数不超过5个**

阿里巴巴《Java开发手册》建议单表索引数量控制在5个以内，这是基于多年实战经验得出的结论。

5个索引对于大多数业务场景已经足够，能够覆盖主要的查询需求。

这个建议不是绝对的，对于特别复杂的业务表可以适当增加，但必须有充分的理由。

规范还建议单个索引的字段数不超过5个，避免创建过于复杂的联合索引。

> 总之： 适当的索引能提高查询效率，过多的索引会影响数据库表的插入和更新功能。

有些时候，不加索引更合适：

- 数据量少的表，不适合加索引
- 更新比较频繁的也不适合加索引

4.2 为什么 阿里巴巴规范建议 是5个？

阿里巴巴的《Java开发手册》建议单表索引不超过5个，为啥呢？

因为，索引 太多的 "副作用" ：

- **写数据变慢**：就像你每写一篇日记，都要在10个不同的目录里更新位置，累不累？
- **占用空间大**：每个索引都要单独存一份数据，就像你为了找书方便，买了10本一模一样的字典放家里
- **MySQL会犯选择困难症**：索引太多，MySQL反而可能选错最快的查询路径
- **维护成本高**：备份、迁移数据时，索引越多越慢

所以，现实中的最佳实践： **5个以内最健康。**

5. 索引过多会 导致的 "七宗罪"

索引虽然能提高查询速度，但过多索引会带来一系列问题，过度索引带来的性能下降和维护困难 ， 这里 总结为索引的"七宗罪"。

理解 "七宗罪" 问题，有助于我们更好地设计索引策略，避免过度索引带来的性能下降和维护困难。

5.1 第一宗罪：写入变慢

每次执行INSERT、UPDATE、DELETE操作时，MySQL不仅要修改数据，还要更新所有相关的索引。

索引越多，写入操作就越慢。

特别是在批量导入数据时，索引会显著降低导入速度。

测试表明，一个没有索引的表可能比有10个索引的表写入速度快10倍以上。

对于在线web服务系统（如电商平台、 金融交易平台），过多的索引会导致系统吞吐量大幅下降。

5.2 第二宗罪：磁盘 空间浪费

占用空间大：每个索引都要单独存一份数据，就像你为了找书方便，买了10本一模一样的字典放家里

每个索引都需要额外的 磁盘 存储空间。

对于InnoDB，索引和数据存储在同一个文件中，索引越多，文件越大。

一个包含10个索引的百万级数据表，索引可能占用几GB甚至更多的空间。

这不仅增加了存储成本，还会影响备份恢复的速度。

5.3 第三宗罪：缓存效率降低

InnoDB使用 Buffer Pool 缓冲池来缓存数据和索引。

索引太多会占用大量 Buffer Pool 缓冲池空间，导致数据和索引的缓存命中率下降。

当 Buffer Pool 缓冲池无法容纳常用数据时，MySQL就需要频繁地从磁盘读取数据，严重影响性能。

合理的索引数量可以让缓冲池缓存更多热点数据。

5.4 第4宗罪：锁竞争加剧

在高并发环境下，索引更新会导致锁竞争加剧。

特别是当多个事务同时修改同一索引时，可能出现锁等待甚至死锁。

InnoDB的行级锁虽然缓解了这个问题，但索引太多仍然会增加锁冲突的概率，影响系统并发性能。

Captured by FireShot Pro: 12 8月 2025, 08:31:20
https://getfireshot.com

5.5 第5宗罪：优化器困惑

MySQL会犯选择困难症：索引太多，MySQL反而可能选错最快的查询路径

当表中有多个索引时，MySQL优化器需要选择使用哪个索引来执行查询。

索引太多会增加优化器做出错误选择的风险，可能导致性能反而下降。

比如优化器可能选择区分度不高的索引，或者错误估计索引的选择性。这时就需要使用FORCE INDEX等提示来强制使用特定索引。

5.6 第6宗罪：维护困难

索引越多，数据库维护工作就越复杂。

ALTER TABLE操作会变得更慢，特别是在大表上添加或删除索引可能需要很长时间。

备份恢复也会变慢，因为需要处理更多的索引数据。

此外，监控和管理大量索引也需要更多的时间和精力。

5.7 第7宗罪：统计信息更新变慢

MySQL使用统计信息来优化查询执行计划。

索引越多，收集和维护统计信息所需的时间和资源就越多。

在数据变化频繁的表上，过时的统计信息可能导致优化器选择低效的执行计划。

虽然可以手动分析表来更新统计信息，但这会增加维护负担。

6. 索引使用实战技巧

掌握了索引的基本原理后，尼恩建议大家 需要了解一些索引的实战技巧， 帮助我们在实际项目中更好地设计和使用索引。

大家对于 索引的使用，存在很多误区，其中 最大的误区是认为"索引越多查询越快"，实际上索引过多会降低整体性能。

另一个误区是为所有查询字段都建索引，这会导致索引泛滥。还有人认为联合索引字段顺序无关紧要，实际上顺序对索引效率影响很大。

此外，过度依赖自动创建的索引、不评估索引使用效果、不删除无用索引等都是常见问题。

6.1 哪些情况不加索引？

第一：数据量小的表（如配置表）不需要索引。为啥呢 ？ 因为数据量小的表 在查询的时候， 全表扫描可能比索引查找更快。

第二：频繁更新的字段（写多读少的字段），要谨慎加索引。为啥呢 ？ 因为每次更新都需要维护索引。

第三：区分度低的字段（如性别、状态标志），通常不适合单独建索引。为啥呢 ？因为索引效果不明显。

第四：太长的字段（如TEXT）不要加索引。如果一定要加，就要使用前缀索引。

第五： NULL值过多的字段，也不建议 加索引。

6.2 如何设计高效索引？

- 首先分析业务查询模式，优先为高频查询条件建索引。
- 联合索引要注意字段顺序，区分度高的字段放前面。
- 避免创建冗余索引，比如已有(a,b)索引就不需要单独建a索引。
- 定期使用EXPLAIN分析慢查询，优化索引策略。
- 考虑使用覆盖索引减少回表操作。
- 对于长字符串，考虑使用前缀索引节省空间。

6.3 对 索引进行 定期监控和优化

索引不是建完就一劳永逸的，需要定期监控和优化。

建议每月至少检查一次索引使用情况，删除无用索引。

> 使用SHOW INDEX FROM table命令可以查看表的索引信息，包括索引名称、字段、基数等。
>
> 通过sys.schema_unused_indexes 视图可以找出长期未使用的索引。
>
> EXPLAIN命令可以分析查询是否使用了合适的索引。

对于数据变化大的表，定期ANALYZE TABLE更新统计信息。

监控索引碎片化程度，必要时重建索引。

建立索引变更评审机制，避免随意添加索引。记录索引变更历史，便于问题追踪。

Captured by FireShot Pro: 12 8月 2025, 08:31:20
https://getfireshot.com

对于重要系统，可以考虑使用索引管理工具。

7. 索引的 真实案例分享

理论结合实践才能更好掌握索引设计，下面分享两个真实案例。

这些案例来自实际项目经验，展示了如何根据具体业务需求设计合理的索引策略，以及不当索引设计可能导致的问题和解决方案。

案例1：电商系统用户表 的索引案例分享

主键使用自增user_id，保证写入性能。

mobile和email字段建立唯一索引，用于登录和密码找回。

register_time建立索引用于新用户分析。

last_login建立索引用于活跃用户统计。

nickname使用前缀索引支持模糊搜索。

避免为gender等低区分度字段单独建索引。

定期清理不活跃用户的索引条目。

案例2：订单系统　订单表 的索引案例分享

主键使用order_id，分布式系统可以考虑雪花ID。

user_id建立索引支持用户查询。

create_time建立索引支持时间范围查询。

status和payment_type建立联合索引用于订单分析。

避免为price等频繁更新的字段单独建索引。

考虑使用部分索引只索引未完成订单。定期归档历史订单减少索引大小。

8. 总结：索引使用黄金法则

经过前面的详细讲解，我们可以总结出一些索引使用的黄金法则。'

记住这些法则可以帮助我们避免常见的索引设计错误，建立高效的数据库结构。

- 不是所有字段都需要索引，只为真正需要的查询条件建索引。
- 联合索引优于多个单列索引，但要注意字段顺序。
- 区分度高的字段更适合索引，低区分度字段考虑联合索引。
- 定期维护比盲目添加更重要，及时删除无用索引。
- 5个以内最健康，超过8个要三思，必须有充分理由。
- 理解业务查询模式是设计好索引的前提。
- 监控和优化是持续过程，不是一次性的工作。

这些法则不是死板的教条，而是指导性的原则，在实际应用中需要根据具体情况进行调整。

遇到问题，找老架构师取经

借助此文的问题 套路 ，大家可以 放手一试，**保证 offer直接到手，还有可能会 涨薪 100%-200%。**

后面，尼恩java面试宝典回录成视频， 给大家打造一套进大厂的塔尖视频。

在面试之前，建议大家系统化的刷一波 5000页《尼恩Java面试宝典PDF》，里边有大量的大厂真题、面试难题、架构难题。

很多小伙伴刷完后， 吊打面试官， 大厂横着走。

在刷题过程中，如果有啥问题，大家可以来 找 40岁老架构师尼恩交流。

另外，如果没有面试机会，**可以找尼恩来改简历、做帮扶。**

遇到职业难题，找老架构取经， 可以省去太多的折腾，省去太多的弯路。

尼恩指导了大量的小伙伴上岸，前段时间，**刚指导 32岁 高中生，冲大厂成功。特批 成为 架构师，年薪 50W，逆天改命 ！！！。**

狠狠卷，实现 “offer自由” 很容易的， 前段时间一个武汉的跟着尼恩卷了2年的小伙伴， 在极度严寒/痛苦被裁的环境下， offer拿到手软， 实现真正的 “offer自由” 。

冲大厂 案例： 全网顶尖、高薪案例， 进大厂拿高薪， 实现薪酬腾飞、人生逆袭

涨一倍：从30万 涨 60万，3年经验小伙 冲大厂成功，逆天了 ！！！

Captured by FireShot Pro: 12 8月 2025, 08:31:20
https://getfireshot.com

📄 **阿里+美团offer**：25岁 屡战屡败 绝望至极。找尼恩转架构升级，1个月拿到阿里+美团offer，逆天改命年薪 50W

📄 **阿里offer**：6年一本 不想 混小厂了。狠卷1年 拿到 得物 + 阿里 offer ， 彻底上岸 ，逆天改命

📄 **字节 offer**：3年经验 ， 足足折腾1年后绝望了，找尼恩陪跑， 2个月 逆天改命 ，拿到 字节&携程 offer

小米offer：📄 7年 普通小二本，冲 小米 成功，年薪60W， 吊打一大票 985/211，狠卷1年，足足涨了50%，人生逆袭

📄 **拼多多offer：2年经验60W，破全网记录**，顶尖案例， 📄 3大厂offer， 双非一本 秒杀985、秒杀211，逆天改命

📄 美团offer：被裁后涨80%，大赚了。从小厂被裁，拿4大厂offer（申通、顺丰、携程、美团），大涨80%，26岁小伙被裁赚翻了

📄 逆天大涨：暴涨200%，29岁/7年/双非一本 ， 从13K涨到 37K ，如何做到的？

📄 逆天改命：27岁被裁2月，转P6降维攻击，2个月提 JD/PDD 两大offer，时来运转，人生翻盘!! 大逆袭!!

📄 急救上岸：29岁（golang）被裁3月，转架构降维打击，收3个大厂offer， 年薪60W，逆天改命

大龄逆袭的案例： 大龄被裁，快速上岸的，远离没有 offer 的焦虑、恐慌

📄 **47岁超级大龄**，被裁员后 找尼恩辅导收 2个offer，一个40多W。 35岁之后，只要技术好，还是有饭吃，关键是找对方向，找对路子

📄 大龄不难：39岁/15年老码农，15天时间40W上岸，管一个team，不用去 铁人三项了！

📄 武汉收5个offer：34岁的CRUD小伙被裁， 尼恩陪跑12天， 收 5个offer，拿到30K，逆涨7K，逆天改命

📄 上岸奇迹：中厂大龄34岁，被裁8月收一大厂offer， 年薪65W，转架构后逆天改命!

100W 年薪 天花板 案例 ，他们 如何 实现薪酬腾飞、人生逆袭？

📄 **年薪100W的底层逻辑**：📄 大厂被裁，他们两个，如何实现年薪百万？

📄 **年薪100W**📄 ：📄 40📄 岁小伙，被裁6个月，猛卷3月，100W逆袭 ，秘诀：升级首席架构/总架构

📄 最新的100W案例：环境太糟，如何升 P8级，年入100W？

职业救助站
实现职业转型，极速上岸

关注**职业救助站**公众号，获取每天职业干货
助您实现**职业转型、职业升级、极速上岸**

技术自由圈
实现架构转型，再无中年危机


关注技术自由圈公众号，获取每天技术千货
一起成为牛逼的未来超级架构师
几十篇架构笔记、5000页面试宝典、20个技术圣经
请加尼恩个人微信 免费拿走
暗号，请在 公众号后台 发送消息：领电子书
如有收获，请点击底部的"在看"和"赞"，谢谢