面试必看！腾讯面试问：MySQL缓存有几级？你能答上来吗？

原创 小北架构师团队 程序员江小北 2025年04月14日 08:30

小北说在前面

> MySQL 有几级缓存？每一级缓存，具体是什么？

最近有同学面试，被问到了这个面试题。 回答的并不好。

因为很多同学在平时对数据库的关注主要在索引、表结构、SQL优化这些方面上，所以会忽略架构层面的东西。

但是没办法，“**上班拧螺丝，面试造火箭**”是面试必须要跨过去的坎。

一位长期做招聘工作的朋友告诉我，其实现在面试，其他都可以不看，你就看一点，他为这次面试做了多少准备。 做的准备越充足，角度越丰富，其实就已经越可以证明他未来的工作潜力。

这证明他是个善于做计划的人，是一个善于学习的人，也是一个肯合作的人，他还是一个能行动的人，这样的人，做什么都行。

所以大家也别抱怨，站在面试官的角度考虑，你就明白了。

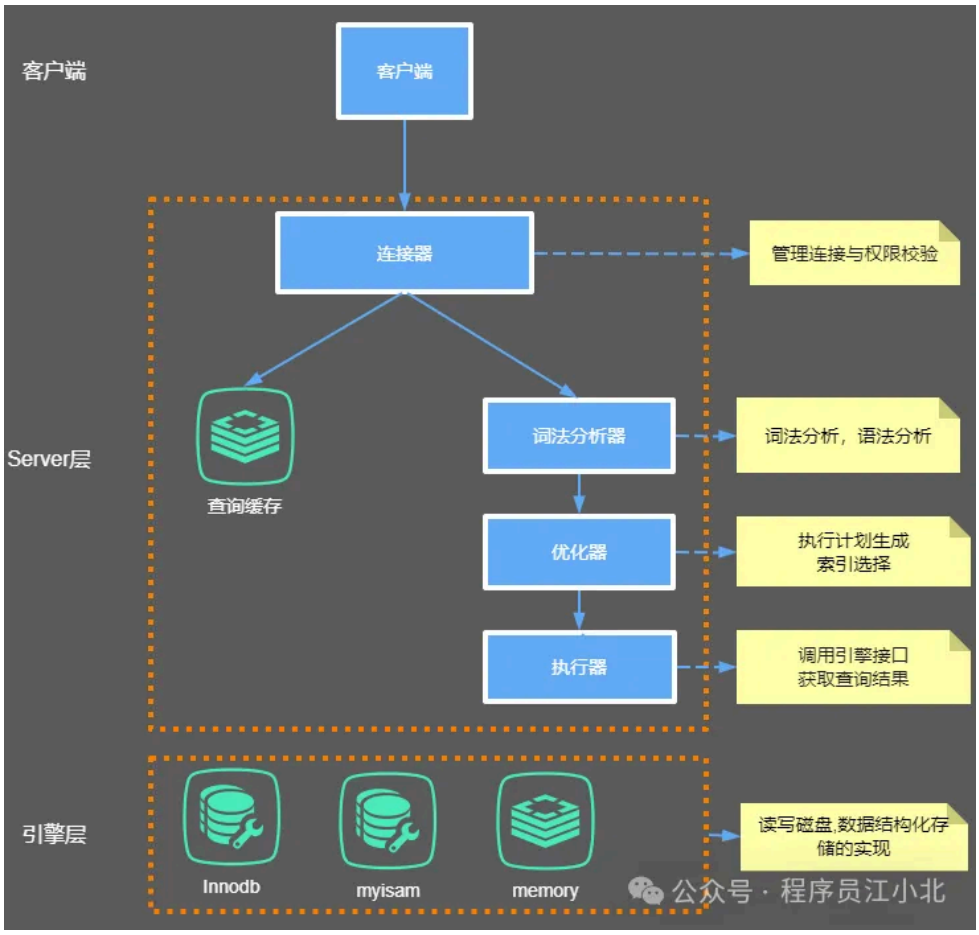
MySQL的缓存机制概览

MySQL的缓存机制主要分为两类：一级缓存和二级缓存。

- **一级缓存**，通常指的是InnoDB缓存，是MySQL中InnoDB存储引擎提供的缓存机制，专门用于存储数据和索引，从而提升数据的访问效率。
- **二级缓存**，即查询缓存（Query Cache），则是MySQL服务器内部的一种缓存机制，专门存储SELECT查询的结果。当相同的查询再次被执行时，系统可以直接从缓存中获取结果，避免重复查询数据库。

二、MySQL 整体架构

Mysql的架构 ，整体是分为服务层、引擎层和文件系统层，其架构图如下所示：



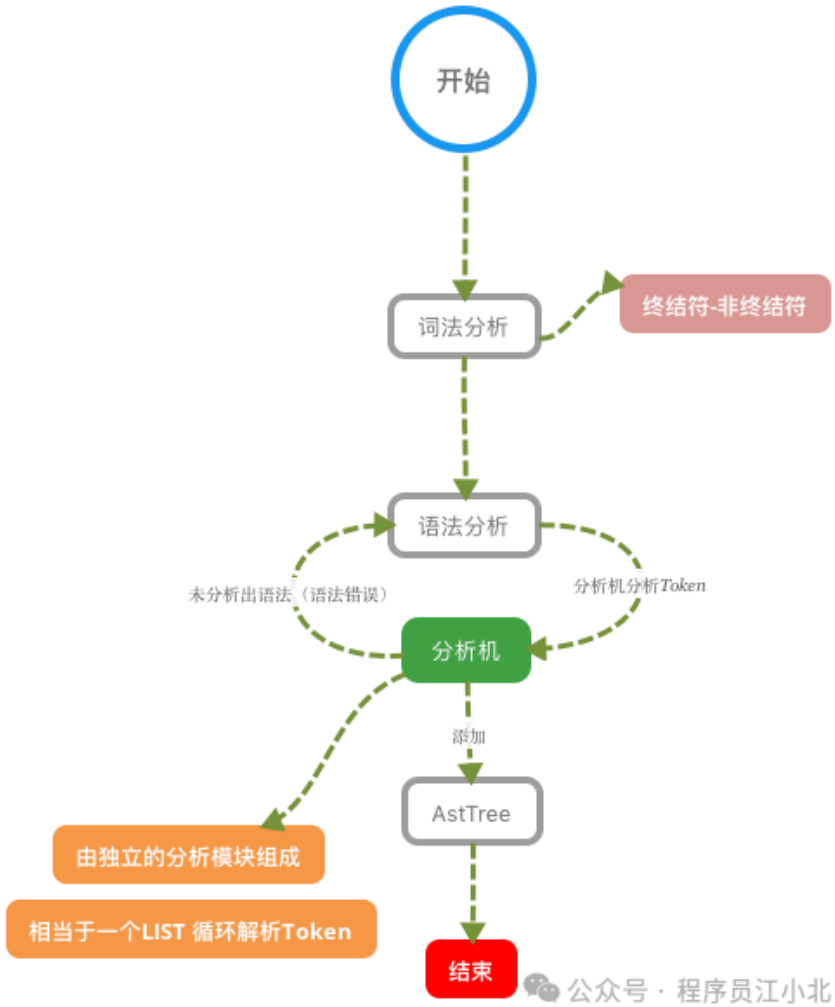
MySQL Server 服务层（Service Layer）解析 SQL 语句、优化查询以及执行操作的，分别有三个关键组件完成：

- **解析器（Parser）**
- **优化器（Optimizer）**
- **执行器（Executor）**。

每个组件在查询执行的过程中扮演不同的角色，下面分别介绍这三者的作用：

解析器（Parser）

在查询执行的初始阶段，解析器承担着将用户提交的 SQL 语句转换为系统可识别结构的任务。



- **词法分析**：系统会将整个 SQL 语句切分成若干标记，这些标记可能是关键字、字段名、表名或运算符等。
- **语法分析**：通过构建解析树的方式，验证 SQL 语句是否符合语法规范，并建立其逻辑结构。
- **语义分析**：进一步校验语句中引用的数据库对象是否存在，同时检查用户是否具有相应的操作权限。

解析的输出结果是一个中间结构，该结构将作为优化器的输入。

优化器（Optimizer）

经过了分析器，MySQL 就知道你要做什么了。在开始执行之前，还要先经过优化器的处理。

> “
>
> 优化器是在表里面有多个索引的时候，决定使用哪个索引；
> 或者在一个语句有多表关联（join）的时候，决定各个表的连接顺序。

比如你执行下面这样的语句，这个语句是执行两个表的 join：

```
mysql> select * from test1 join test2 using(ID) where test1.name=yyyy and test2.name=xxx;
```

既可以先从表 test1 里面取出 name=yyyy ID 值，再根据 ID 值关联到表 test2，再判断 test2 里面 name的值是否等于 yyyy。 也可以先从表 test2 里面取出 name=xxx 的记录的 ID 值，再根据 ID 值关联到 test1，再判断 test1 里面 name的值是否等于 yyyy。

这两种执行方法的逻辑结果是一样的，但是执行的效率会有不同，而优化器的作用就是决定选择使用哪一个方案。

优化器阶段完成后，这个语句的执行方案就确定下来了，然后进入执行器阶段。如果你还有一些疑问，比如优化器是怎么选择索引的，有没有可能选择错等等

总结下来就是：优化器的作用在于选择出成本最低、效率最优的执行策略，以提升查询性能。

执行器（Executor）

执行器负责将优化器生成的执行计划付诸实践，完成数据的实际访问与结果的生成。

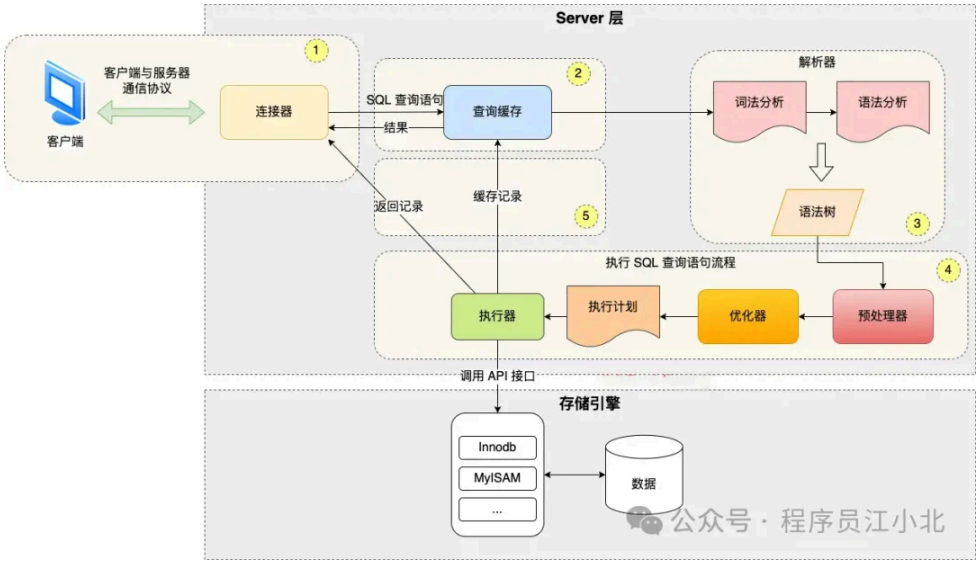
- **权限检查**：执行前，系统会验证当前用户是否具备执行该操作的权限，如权限不足则中止操作并返回错误信息。
- **执行计划调度**：根据执行计划的步骤，执行器逐条调度存储引擎接口完成操作。对于查询，系统读取所需数据；对于插入或更新，则写入或修改数据。
- **结果返回**：完成数据访问后，执行器将结果组织为客户端可识别的格式进行输出。在多步骤操作中（如连接查询），执行器负责协调整体流程，合成最终的结果集。

核心组件的交互流程

1、 **解析器**将用户提交的SQL语句构建为解析树；
2、 **优化器**在解析树基础上生成最优的执行计划；
3、 **执行器**依据计划逐步调用存储引擎接口，完成查询并返回结果；

解析器、优化器与执行器之间紧密配合，构成从接收请求到返回结果的完整执行链条。

一条完整的SQL查询执行流程



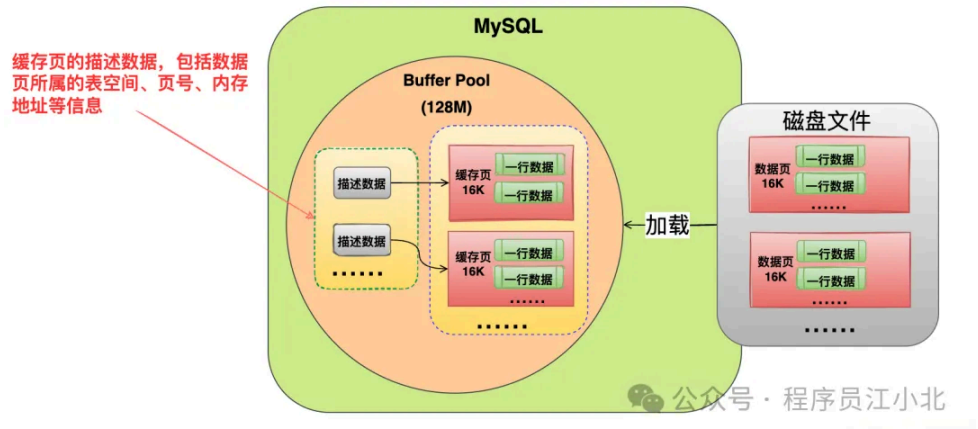
一条SQL查询语句的执行从发起到返回结果，涉及多个步骤和系统组件。

执行过程中，可能会用到缓存来优化性能。

以下是执行流程的概述，以及缓存的可能应用：

1、 客户端请求：客户端（例如应用程序）向MySQL服务器发送SQL查询请求；
2、 解析器：MySQL服务器接收到SQL查询后，首先由解析器进行解析，检查SQL语句的语法是否符合规则；
3、 优化器：解析后，优化器根据查询语句以及数据库的元数据（如表结构、索引等）生成一个或多个执行计划；
4、 权限检查：MySQL服务器检查执行查询的用户是否拥有足够的权限；
5、 缓存查询：若查询能够被缓存，MySQL会检查一级缓存（如InnoDB缓冲池）或二级缓存（在查询缓存启用的情况下）。
二级缓存（查询缓存，MySQL 8.0之前）：如果查询缓存启用，服务器会检查查询缓存是否已有该查询的结果，若有，则直接返回缓存结果；
6、 执行器：若查询未在缓存中找到，执行器根据优化器生成的执行计划执行查询。执行过程中，可能会涉及到数据的读取与写入，缓存（如InnoDB缓冲池）会更新。

一级缓存（InnoDB缓冲池）：若查询所需的数据或索引已经存在于缓冲池中，则无需访问磁盘，直接从缓冲池中获取数据。



7、 返回结果：查询执行完毕后，结果集被返回给客户端；
8、 更新缓存：对于写操作（如INSERT、UPDATE、DELETE），相关缓存（一级缓存和查询缓存）将更新或失效，确保数据的一致性；
9、 日志记录：MySQL会记录查询日志（例如慢查询日志）以供后续性能分析；
10、 关闭连接：查询完成后，客户端可以选择关闭与MySQL服务器的连接，或保持连接以便于后续查询。

需要注意的是，从MySQL 8.0版本开始，查询缓存已被移除，因此在使用MySQL 8.0及以上版本时，查询缓存不会被使用。

此外，二级缓存（如InnoDB缓冲池）的使用由InnoDB存储引擎自动管理，无需用户干预。用户可以通过调整缓冲池大小及其他相关参数来优化性能。

MySQL 8.0为何移除查询二级缓存机制

在MySQL 8.0中，查询缓存（Query Cache）被完全移除。

早期版本中，查询缓存被设计为一种提升重复查询效率的机制，通过将SELECT查询的结果缓存在内存中，以避免重复执行相同查询。然而在实际应用中，查询缓存常常成为性能瓶颈，尤其是在并发高或写操作频繁的系统中，缓存内容会因表更新而频繁失效，导致命中率下降，不仅无法提升性能，反而增加了锁争用和管理开销。

为了提升系统整体的可扩展性和性能稳定性，MySQL 8.0开始完全摒弃查询缓存，转而推荐使用更加灵活和高效的缓存策略，例如：

1、 应用层缓存：通过在应用中集成缓存逻辑，结合如 Redis、Memcached 等系统，对热点数据进行缓存，提高访问效率；
2、 持久化存储引擎缓存：借助InnoDB的缓冲池机制，对数据页和索引页进行高效缓存，由存储引擎自动管理；
3、 替代存储引擎：在特定场景下，仍可使用如MyISAM这类支持查询缓存的引擎，但需注意其不支持事务等局限性；

MySQL 8.0通过强化存储引擎的内部缓存机制、改进索引结构以及提供更多性能调优工具，来应对高并发查询带来的挑战，从而不再依赖传统的查询缓存功能。

MySQL 8.0之前的查询缓存配置方法

在MySQL 8.0版本之前，可以通过系统变量配置查询缓存机制，以提升相同查询语句的执行效率。

配置过程相对简单，主要包括开启缓存功能、设置缓存大小以及执行查询语句。

步骤	描述	代码
1	启用查询缓存	`SET GLOBAL query_cache_type = 1;`
2	设置查询缓存大小	`SET GLOBAL query_cache_size = 64 * 1024 * 1024;`
3	执行查询	`SELECT * FROM table_name;`

配置说明

- `SET GLOBAL query_cache_type = 1;` 用于启用查询缓存功能。设置为 `1` 表示开启；
- `SET GLOBAL query_cache_size = 64 * 1024 * 1024;` 用于设置查询缓存的容量，单位为字节，此处设置为64MB；
- `SELECT * FROM table_name;` 执行查询语句。如果缓存中已存在该查询的结果集，将直接返回缓存内容，无需重新执行查询逻辑。

这种查询缓存机制适用于读操作频繁、写操作较少的场景。在执行相同查询语句时可以显著减少响应时间。然而，需要注意的是，每当相关表发生更新时，缓存会自动失效，这也是后来该功能被逐步淘汰的重要原因之一。

应用层缓存作为外置缓存的替代方案

由于MySQL 8.0版本已经不再支持查询缓存，开发者在处理频繁访问的数据时，需要引入更灵活高效的缓存机制。外置缓存是目前广泛应用的解决方案，尤其适用于对性能要求较高的业务场景。

常见的方式包括使用Redis、Memcached等缓存系统，在应用层实现对查询结果的缓存，从而减轻数据库压力，提高系统响应速度。

- **应用层缓存**：通过在应用逻辑中加入缓存控制，将查询结果缓存在Redis或Memcached中，适合高频率访问的热点数据。缓存的更新和失效策略通常由开发者自定义控制。
- **本地缓存**：对于极高频访问且数据量较小的热点数据，可以在本地内存中进行缓存，减少网络开销，进一步提升读取性能。

通过引入这些机制，系统能够更好地应对高并发访问场景，在确保数据一致性的基础上，实现更优的查询响应体验。

小北架构团队的塔尖 SQL 面试题

count(*)、count(1)哪个更快？面试必问：通宵整理的十道经典MySQL必问面试题

被问懵了？MySQL 面试：DISTINCT 和 GROUP BY 效率到底谁更高？

面试官问：MySQL自增ID用完了，怎么办？

面试必问：MySQL死锁 是什么，如何解决？（史上最全）

亿级电商流量，高并发下Redis与MySQL的数据一致性如何保证

腾讯二面：1.2 亿级大表, 如何 加索引？

大厂都在用的分布式事务方案，Seata + RocketMQ带你打破 10万 QPS瓶颈

说在最后

学习可以是功利性质的（当然也可以是因为纯粹的热爱），如果不是为了高薪，我们又何苦学这么多呢？

准备升职加薪/跳槽的同学，那就一定要认真的学习，谁知道哪次面试遇到了，可能你就没有把握住。

如有收获，请点击底部的"在看"和"赞"，谢谢

小北私藏精品　热门推荐

小北联合公司合伙人，一线大厂在职架构师耗时9个月联合打造了**《2024年Java高级架构师课程》**本课程对标外面3万左右的架构培训课程，分10个阶段，目前已经更新了**181G**视频，已经更新**1000+**个小时视频，**一次购买，持续更新，无需2次付费**

第3版：互联网大厂面试题

包括 Java 集合、JVM、多线程、并发编程、设计模式、算法调优、Spring全家桶、Java、MyBatis、ZooKeeper、Dubbo、Elasticsearch、Memcached、MongoDB、Redis、MySQL、RabbitMQ、Kafka、Linux、Netty、Tomcat、Python、HTML、CSS、Vue、React、JavaScript、Android 大数据、阿里巴巴等大厂面试题等、等技术栈！

阅读原文： 高清 7701页大厂面试题 PDF

阅读原文