03丨学会用数据库的方式思考SQL是如何执行的

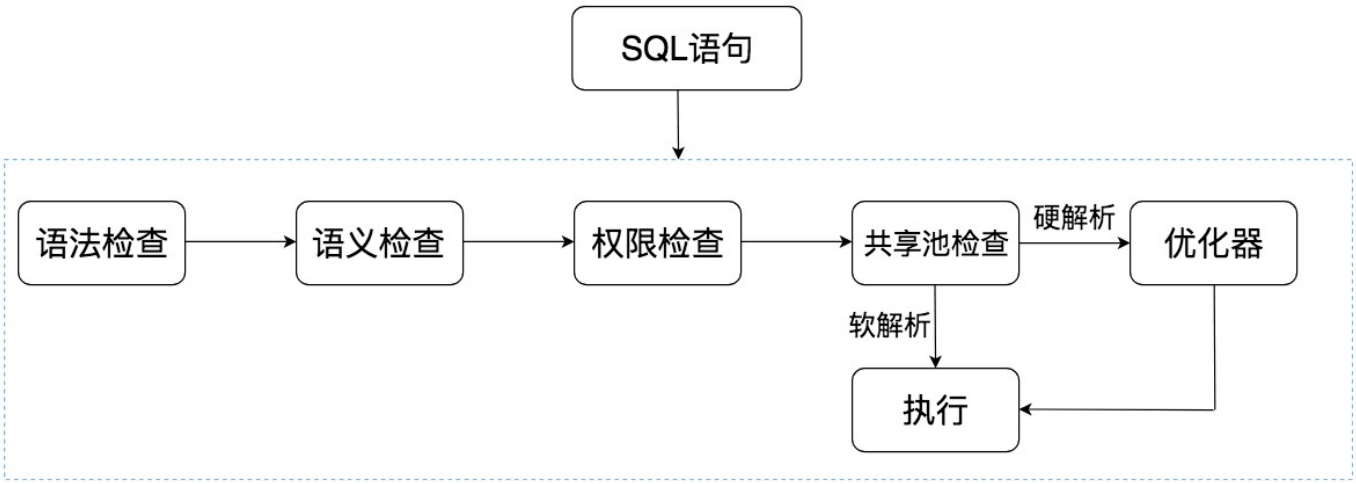
通过上一篇文章对不同的 DBMS 的介绍，你应该对它们有了一些基础的了解。虽然 SQL 是声明式语言，我们可以像使用英语一样使用它，不过在 RDBMS（关系型数据库管理系统）中，SQL 的实现方式还是有差别的。今天我们就从数据库的角度来思考一下 SQL 是如何被执行的。

关于今天的内容，你会从以下几个方面进行学习：

1. Oracle 中的 SQL 是如何执行的，什么是硬解析和软解析；
2. MySQL 中的 SQL 是如何执行的，MySQL 的体系结构又是怎样的；
3. 什么是存储引擎，MySQL 的存储引擎都有哪些？

Oracle 中的 SQL 是如何执行的

我们先来看下 SQL 在 Oracle 中的执行过程：



从上面这张图中可以看出，SQL 语句在 Oracle 中经历了以下的几个步骤。

1. 语法检查：检查 SQL 拼写是否正确，如果不正确，Oracle 会报语法错误。
2. 语义检查：检查 SQL 中的访问对象是否存在。比如我们在写 SELECT 语句的时候，列名写错了，系统就会提示错误。语法检查和语义检查的作用是保证 SQL 语句没有错误。
3. 权限检查：看用户是否具备访问该数据的权限。
4. 共享池检查：共享池（Shared Pool）是一块内存池，最主要的作用是缓存 SQL 语句和该语句的执行计划。Oracle 通过检查共享池是否存在 SQL 语句的执行计划，来判断

进行软解析，还是硬解析。那软解析和硬解析又该怎么理解呢？
在共享池中，Oracle 首先对 SQL 语句进行 Hash 运算，然后根据 Hash 值在库缓存（Library Cache）中查找，如果存在 SQL 语句的执行计划，就直接拿来执行，直接进入“执行器”的环节，这就是软解析。
如果没有找到 SQL 语句和执行计划，Oracle 就需要创建解析树进行解析，生成执行计划，进入“优化器”这个步骤，这就是硬解析。

5. 优化器：优化器中就是要进行硬解析，也就是决定怎么做，比如创建解析树，生成执行计划。
6. 执行器：当有了解析树和执行计划之后，就知道了 SQL 该怎么被执行，这样就可以在执行器中执行语句了。

共享池是 Oracle 中的术语，包括了库缓存，数据字典缓冲区等。我们上面已经讲到了库缓存区，它主要缓存 SQL 语句和执行计划。而数据字典缓冲区存储的是 Oracle 中的对象定义，比如表、视图、索引等对象。当对 SQL 语句进行解析的时候，如果需要相关的数据，会从数据字典缓冲区中提取。

库缓存这一个步骤，决定了 SQL 语句是否需要进行硬解析。为了提升 SQL 的执行效率，我们应该尽量避免硬解析，因为在 SQL 的执行过程中，创建解析树，生成执行计划是很消耗资源的。

你可能会问，如何避免硬解析，尽量使用软解析呢？在 Oracle 中，绑定变量是它的一大特色。绑定变量就是在 SQL 语句中使用变量，通过不同的变量取值来改变 SQL 的执行结果。这样做的好处是能提升软解析的可能性，不足之处在于可能会导致生成的执行计划不够优化，因此是否需要绑定变量还需要视情况而定。

举个例子，我们可以使用下面的查询语句：

复制代码

```
SQL> select * from player where player_id = 10001;
```

你也可以使用绑定变量，如：

复制代码

```
SQL> select * from player where player_id = :player_id;
```

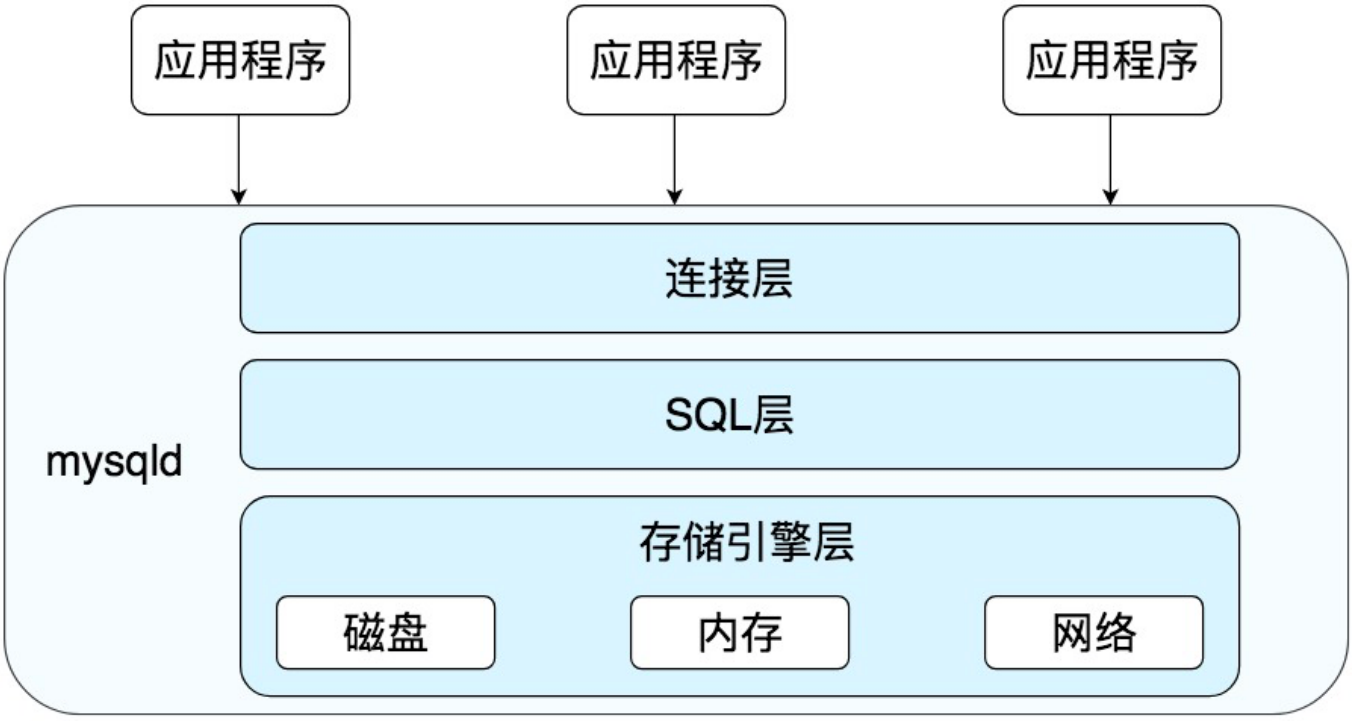
这两个查询语句的效率在 Oracle 中是完全不同的。如果你在查询 player_id = 10001 之后，还会查询 10002、10003 之类的数据，那么每一次查询都会创建一个新的查询解析。而第二种方式使用了绑定变量，那么在第一次查询之后，在共享池中就会存在这类查询的执行计划，也就是软解析。

因此我们可以通过使用绑定变量来减少硬解析，减少 Oracle 的解析工作量。但是这种方式也有缺点，使用动态 SQL 的方式，因为参数不同，会导致 SQL 的执行效率不同，同时 SQL 优化也会比较困难。

MySQL 中的 SQL 是如何执行的

Oracle 中采用了共享池来判断 SQL 语句是否存在缓存和执行计划，通过这一步骤我们可以知道应该采用硬解析还是软解析。那么在 MySQL 中，SQL 是如何被执行的呢？

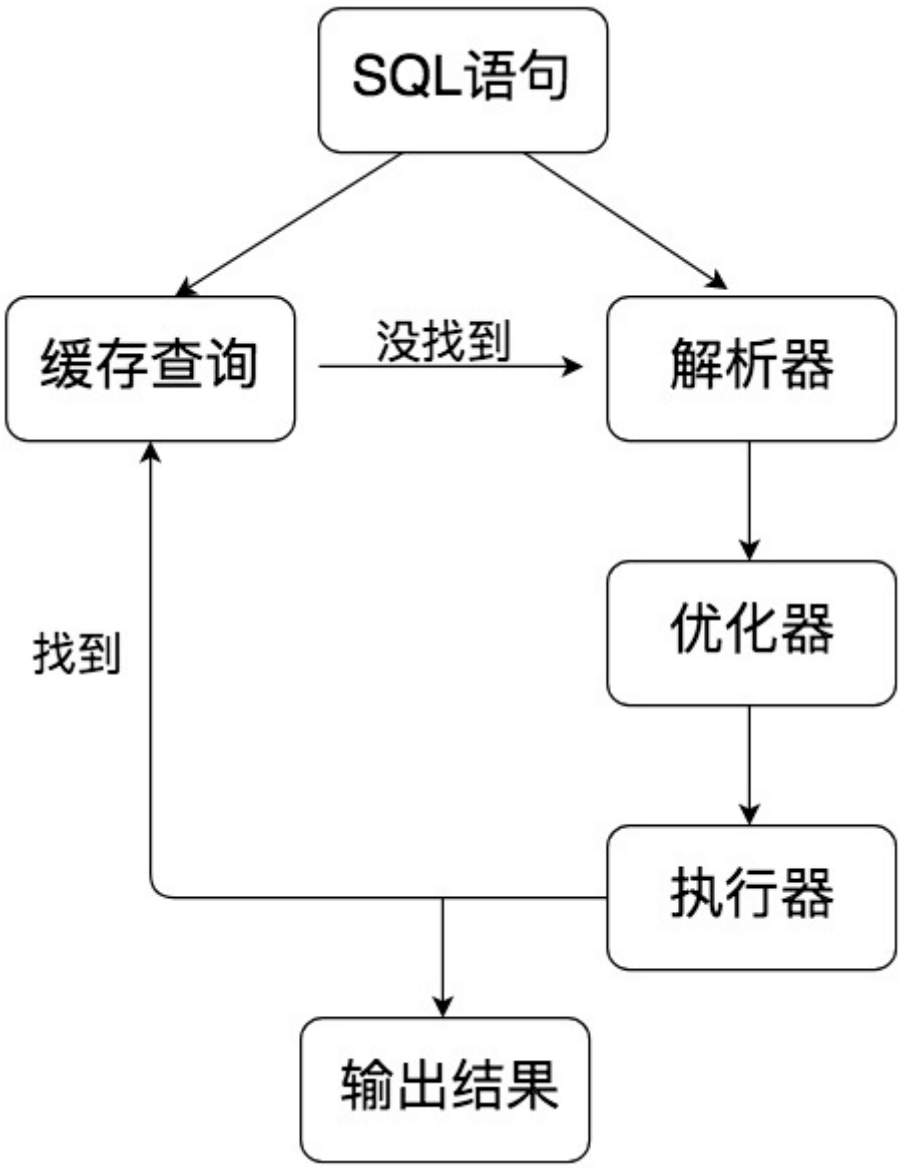
首先 MySQL 是典型的 C/S 架构，即 Client/Server 架构，服务器端程序使用的 mysqld。整体的 MySQL 流程如下图所示：



你能看到 MySQL 由三层组成：

1. 连接层：客户端和服务器端建立连接，客户端发送 SQL 至服务器端；
2. SQL 层：对 SQL 语句进行查询处理；
3. 存储引擎层：与数据库文件打交道，负责数据的存储和读取。

其中 SQL 层与数据库文件的存储方式无关，我们来看下 SQL 层的结构：



1. 查询缓存：Server 如果在查询缓存中发现了这条 SQL 语句，就会直接将结果返回给客户端；如果没有，就进入到解析器阶段。需要说明的是，因为查询缓存往往效率不高，所以在 MySQL8.0 之后就抛弃了这个功能。
2. 解析器：在解析器中对 SQL 语句进行语法分析、语义分析。
3. 优化器：在优化器中会确定 SQL 语句的执行路径，比如是根据全表检索，还是根据索引来检索等。
4. 执行器：在执行之前需要判断该用户是否具备权限，如果具备权限就执行 SQL 查询并返回结果。在 MySQL8.0 以下的版本，如果设置了查询缓存，这时会将查询结果进行缓存。

你能看到 SQL 语句在 MySQL 中的流程是：SQL 语句→缓存查询→解析器→优化器→执行器。在一部分中，MySQL 和 Oracle 执行 SQL 的原理是一样的。

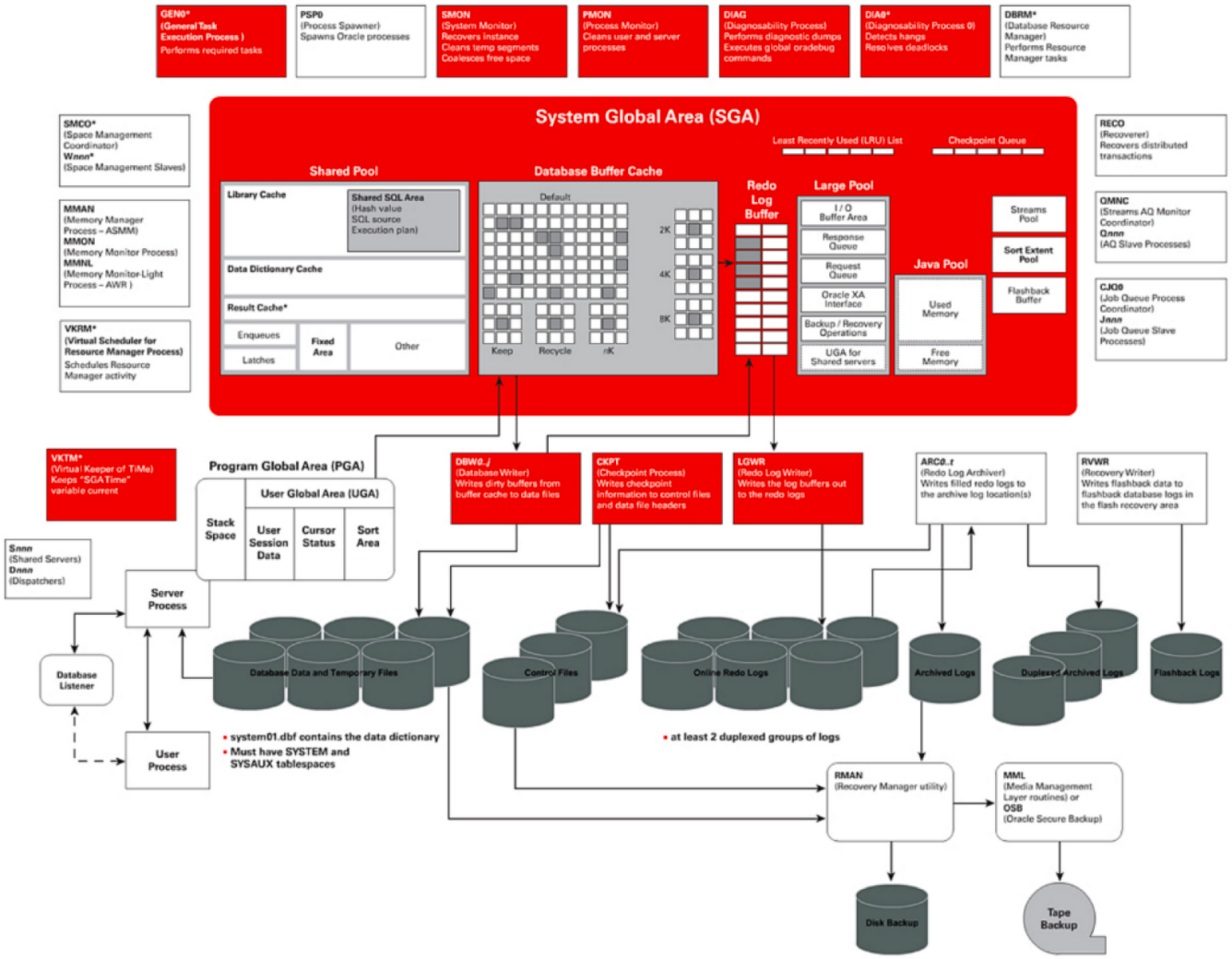
与 Oracle 不同的是，MySQL 的存储引擎采用了插件的形式，每个存储引擎都面向一种特定的数据库应用环境。同时开源的 MySQL 还允许开发人员设置自己的存储引擎，下面是一些常见的存储引擎：

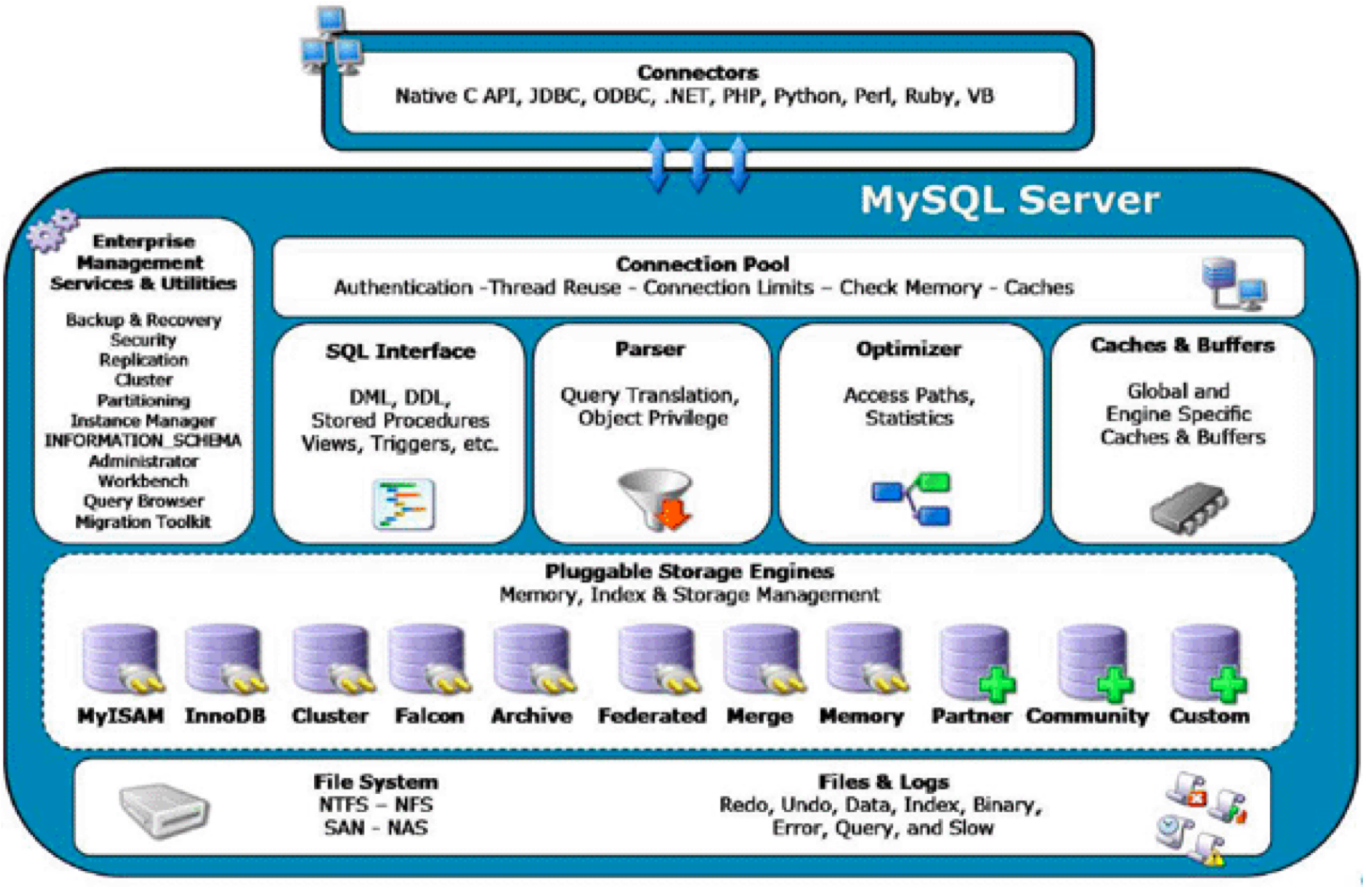
1. InnoDB 存储引擎：它是 MySQL 5.5 版本之后默认的存储引擎，最大的特点是支持事务、行级锁定、外键约束等。
2. MyISAM 存储引擎：在 MySQL 5.5 版本之前是默认的存储引擎，不支持事务，也不支持外键，最大的特点是速度快，占用资源少。
3. Memory 存储引擎：使用系统内存作为存储介质，以便得到更快的响应速度。不过如果 mysqld 进程崩溃，则会导致所有的数据丢失，因此我们只有当数据是临时的情况下才使用 Memory 存储引擎。
4. NDB 存储引擎：也叫做 NDB Cluster 存储引擎，主要用于 MySQL Cluster 分布式集群环境，类似于 Oracle 的 RAC 集群。
5. Archive 存储引擎：它有很好的压缩机制，用于文件归档，在请求写入时会进行压缩，所以也经常用来做仓库。

需要注意的是，数据库的设计在于表的设计，而在 MySQL 中每个表的设计都可以采用不同的存储引擎，我们可以根据实际的数据处理需要来选择存储引擎，这也是 MySQL 的强大之处。

数据库管理系统也是一种软件

我们刚才了解了 SQL 语句在 Oracle 和 MySQL 中的执行流程，实际上完整的 Oracle 和 MySQL 结构图要复杂得多：


GEN0*
(General Task Execution Process)
Performs required tasks
PSP0
(Process Spawner)
Spawns Oracle processes
SMON
(System Monitor)
Recovers instance
Cleans temp segments
Coalesces free space
PMON
(Process Monitor)
Cleans user and server processes
DIAG
(Diagnosability Process)
Performs diagnostic dumps
Executes global oradebug commands
DIAG*
(Diagnosability Process 0)
Detects hangs
Resolves deadlocks
DBRM*
(Database Resource Manager)
Performs Resource Manager tasks
System Global Area (SGA)
Least Recently Used (LRU) List
Checkpoint Queue
Shared Pool
Library Cache
Shared SQL Area
(Hash value
SQL source
Execution plan)
Data Dictionary Cache
Result Cache*
Enqueues
Latches
Fixed Area
Other
Database Buffer Cache
Default
2K
4K
8K
Keep
Recycle
nK
Redo Log Buffer
Large Pool
I / O Buffer Area
Response Queue
Request Queue
Oracle XA Interface
Backup / Recovery Operations
UGA for Shared servers
Java Pool
Used Memory
Free Memory
Streams Pool
Sort Extent Pool
Flashback Buffer
SMCO*
(Space Management Coordinator)
Wnnn*
(Space Management Slaves)
MMAN
(Memory Manager Process – ASMM)
MMON
(Memory Monitor Process)
MMNL
(Memory Monitor Light Process – AWR)
VKRM*
(Virtual Scheduler for Resource Manager Process)
Schedules Resource Manager activity
RECO
(Recoverer)
Recovers distributed transactions
QMNC
(Streams AQ Monitor Coordinator)
Qnnn
(AQ Slave Processes)
CJQ0
(Job Queue Process Coordinator)
Jnnn
(Job Queue Slave Processes)
VKTM*
(Virtual Keeper of TiMe)
Keeps "SGA Time" variable current
Program Global Area (PGA)
User Global Area (UGA)
Stack Space
User Session Data
Cursor Status
Sort Area
DBW0..j
(Database Writer)
Writes dirty buffers from buffer cache to data files
CKPT
(Checkpoint Process)
Writes checkpoint information to control files and data file headers
LGWR
(Redo Log Writer)
Writes the log buffers out to the redo logs
ARC0..t
(Redo Log Archiver)
Writes filled redo logs to the archive log location(s)
RVWR
(Recovery Writer)
Writes flashback data to flashback database logs in the flash recovery area
Snnn
(Shared Servers)
Dnnn
(Dispatchers)
Server Process
Database Listener
User Process
Database Data and Temporary Files
Control Files
Online Redo Logs
Archived Logs
Duplexed Archived Logs
Flashback Logs
• system01.dbf contains the data dictionary
• Must have SYSTEM and SYSAUX tablespaces
• at least 2 duplexed groups of logs
RMAN
(Recovery Manager utility)
MML
(Media Management Layer routines) or
OSB
(Oracle Secure Backup)
Disk Backup
Tape Backup


Connectors
Native C API, JDBC, ODBC, .NET, PHP, Python, Perl, Ruby, VB
MySQL Server
Enterprise Management Services & Utilities
Backup & Recovery
Security
Replication
Cluster
Partitioning
Instance Manager
INFORMATION_SCHEMA
Administrator
Workbench
Query Browser
Migration Toolkit
Connection Pool
Authentication -Thread Reuse - Connection Limits – Check Memory - Caches
SQL Interface
DML, DDL,
Stored Procedures
Views, Triggers, etc.
Parser
Query Translation,
Object Privilege
Optimizer
Access Paths,
Statistics
Caches & Buffers
Global and
Engine Specific
Caches & Buffers
Pluggable Storage Engines
Memory, Index & Storage Management
MyISAM
InnoDB
Cluster
Falcon
Archive
Federated
Merge
Memory
Partner
Community
Custom
File System
NTFS – NFS
SAN - NAS
Files & Logs
Redo, Undo, Data, Index, Binary,
Error, Query, and Slow

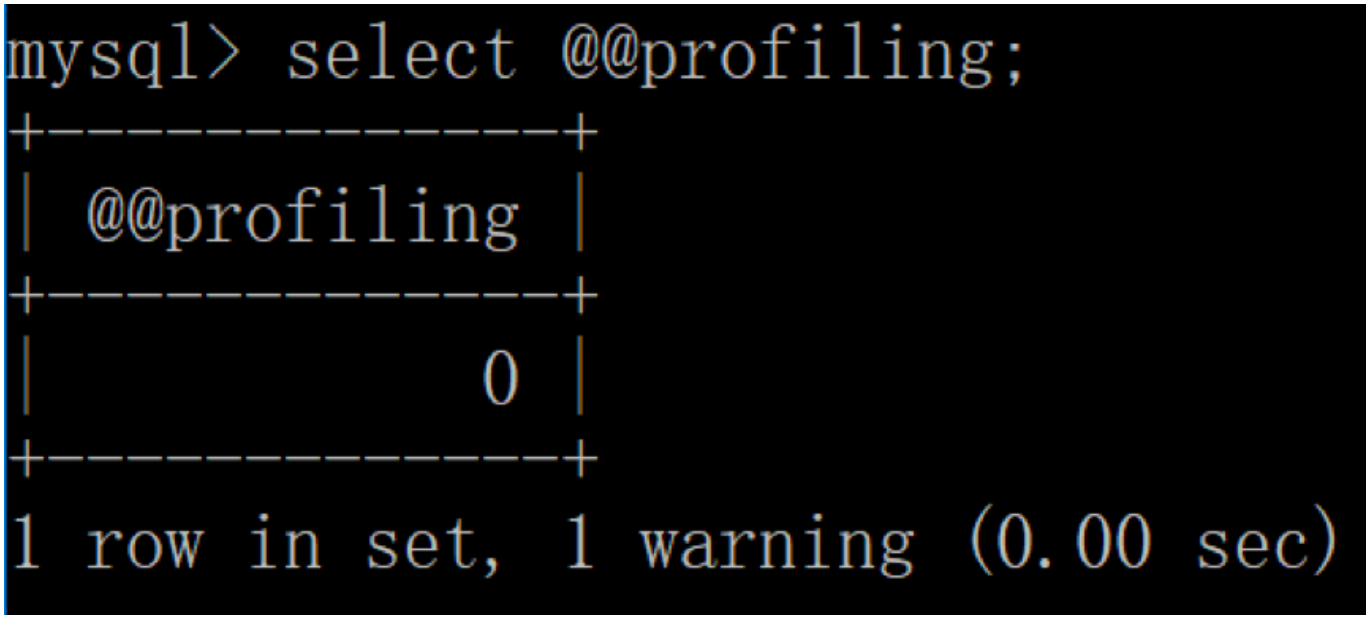
如果你只是简单地把 MySQL 和 Oracle 看成数据库管理系统软件，从外部看难免会觉得“晦涩难懂”，毕竟组织结构太多了。我们在学习的时候，还需要具备抽象的能力，抓取最核心的部分：SQL 的执行原理。因为不同的 DBMS 的 SQL 的执行原理是相通的，只是在不同的软件中，各有各的实现路径。

既然一条 SQL 语句会经历不同的模块，那我们就来看下，在不同的模块中，SQL 执行所使用的资源（时间）是怎样的。下面我来教你如何在 MySQL 中对一条 SQL 语句的执行时间进行分析。

首先我们需要看下 profiling 是否开启，开启它可以让 MySQL 收集在 SQL 执行时所使用的资源情况，命令如下：

复制代码

```
1 mysql> select @@profiling;
```



profiling=0 代表关闭，我们需要把 profiling 打开，即设置为 1：

复制代码

```
1 mysql> set profiling=1;
```

然后我们执行一个 SQL 查询（你可以执行任何一个 SQL 查询）：

复制代码

```
mysql> select * from wucai.heros;
```

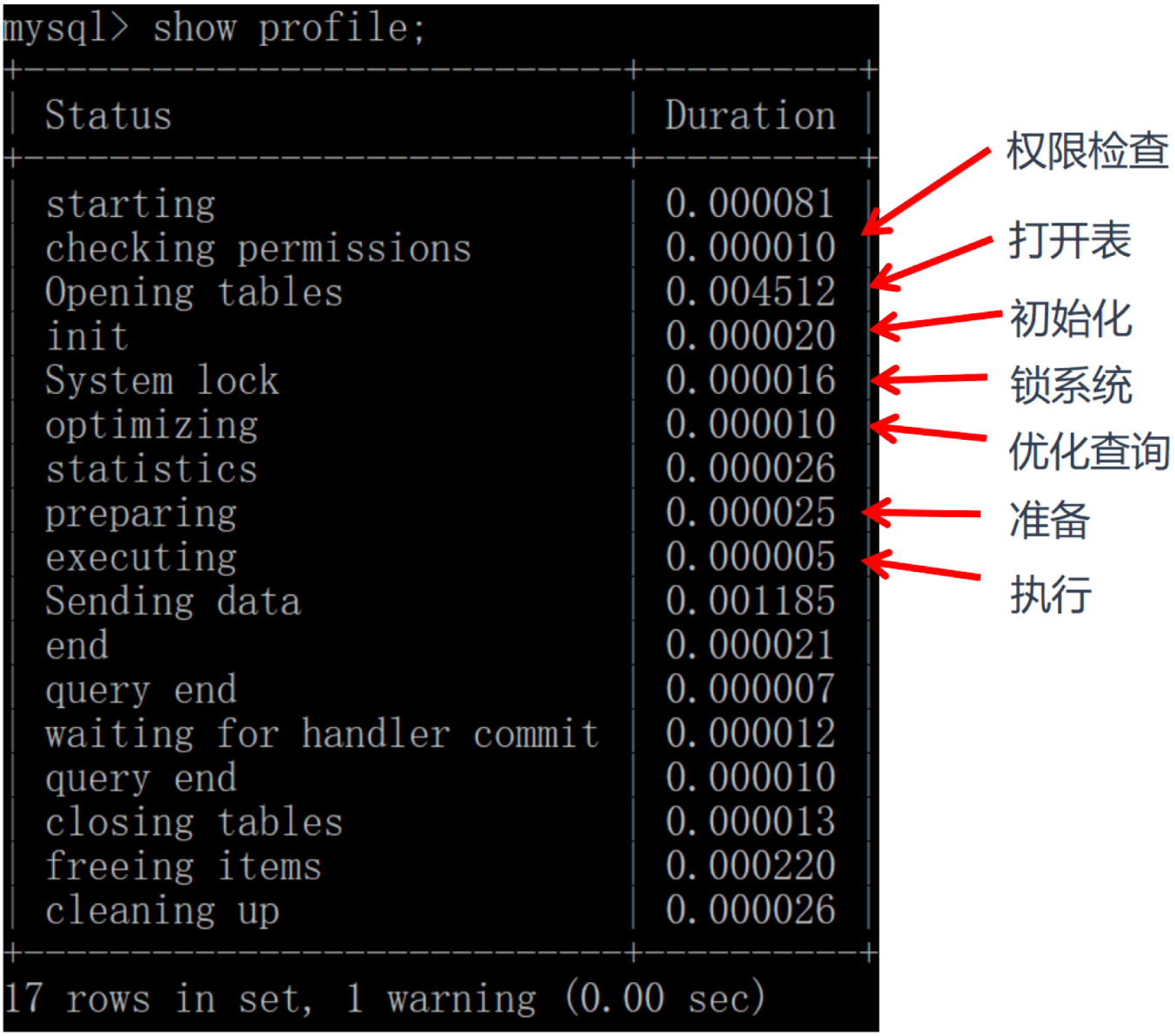
查看当前会话所产生的所有 profiles：

```
mysql> show profiles;
+----------+------------+---------------------------+
| Query_ID | Duration   | Query                     |
+----------+------------+---------------------------+
|        1 | 0.00064700 | select @@profiling        |
|        2 | 0.00619700 | select * from wucai.heros |
+----------+------------+---------------------------+
2 rows in set, 1 warning (0.00 sec)
```

你会发现我们刚才执行了两次查询，Query ID 分别为 1 和 2。如果我们想要获取上一次查询的执行时间，可以使用：

复制代码

```
mysql> show profile;
```



当然你也可以查询指定的 Query ID，比如：

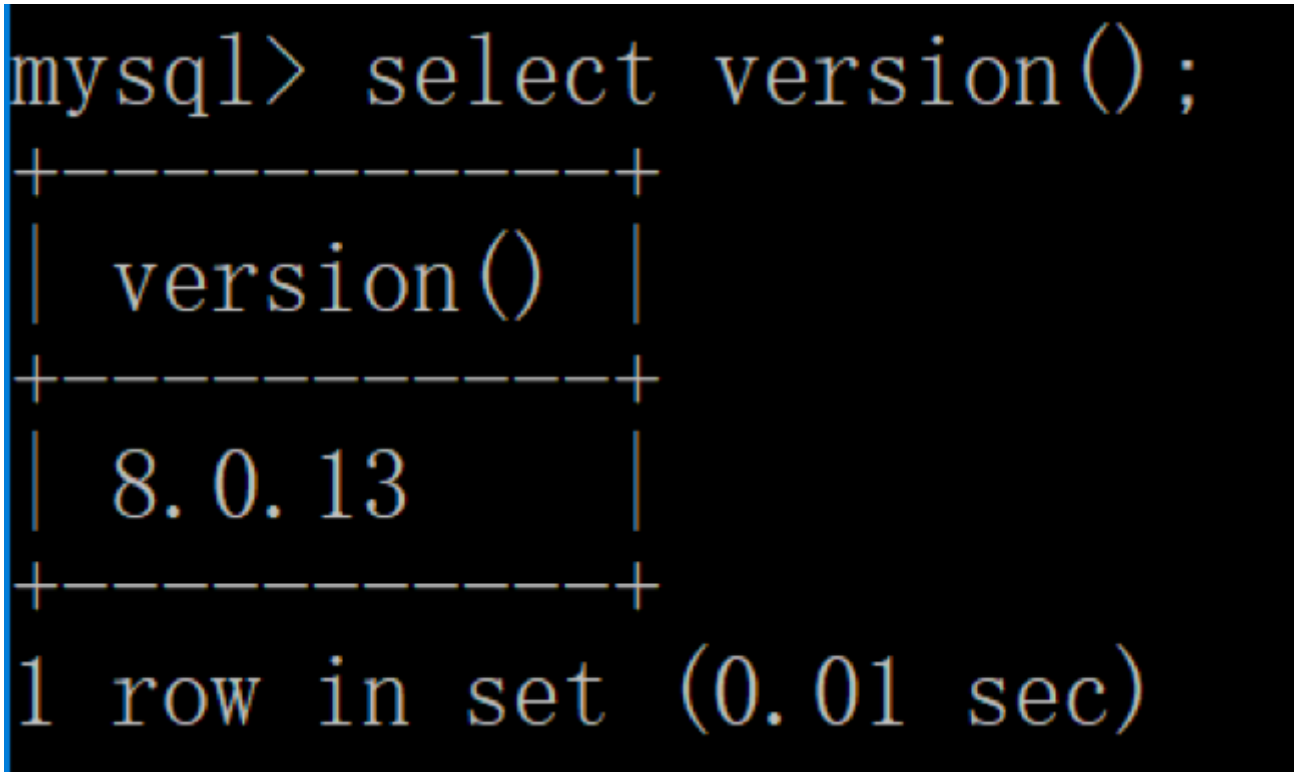
复制代码

```
1 mysql> show profile for query 2;
```

查询 SQL 的执行时间结果和上面是一样的。

在 8.0 版本之后，MySQL 不再支持缓存的查询，原因我在上文已经说过。一旦数据表有更新，缓存都将清空，因此只有数据表是静态的时候，或者数据表很少发生变化时，使用缓存查询才有价值，否则如果数据表经常更新，反而增加了 SQL 的查询时间。

你可以使用 select version() 来查看 MySQL 的版本情况。

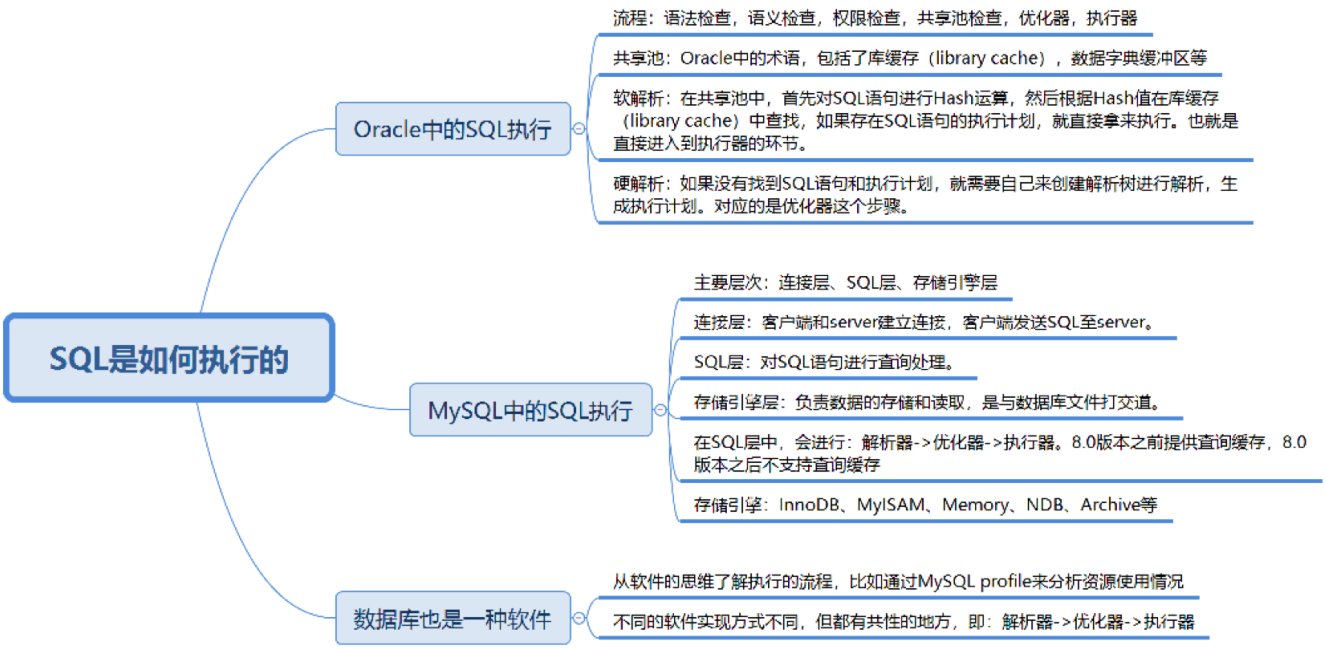


总结

我们在使用 SQL 的时候，往往只见树木，不见森林，不会注意到它在各种数据库软件中是如何执行的，今天我们从全貌的角度来理解这个问题。你能看到不同的 RDBMS 之间有相同的地方，也有不同的地方。

相同的地方在于 Oracle 和 MySQL 都是通过解析器→优化器→执行器这样的流程来执行 SQL 的。

但 Oracle 和 MySQL 在进行 SQL 的查询上面有软件实现层面的差异。Oracle 提出了共享池的概念，通过共享池来判断是进行软解析，还是硬解析。而在 MySQL 中，8.0 以后的版本不再支持查询缓存，而是直接执行解析器→优化器→执行器的流程，这一点从 MySQL 中的 show profile 里也能看到。同时 MySQL 的一大特色就是提供了各种存储引擎以供选择，不同的存储引擎有各自的使用场景，我们可以针对每张表选择适合的存储引擎。



今天的内容到这里就结束了，你能说一下 Oracle 中的绑定变量是什么，使用它有什么优缺点吗？MySQL 的存储引擎是一大特色，其中 MyISAM 和 InnoDB 都是常用的存储引擎，这两个存储引擎的特性和使用场景分别是什么？

最后留一道选择题吧，解析后的 SQL 语句在 Oracle 的哪个区域中进行缓存？

A. 数据缓冲区
B. 日志缓冲区
C. 共享池
D. 大池

欢迎你在评论区写下你的思考，我会在评论区与你一起交流，如果这篇文章帮你理顺了 Oracle 和 MySQL 执行 SQL 的过程，欢迎你把它分享给你的朋友或者同事。