第三讲：MySQL 系统架构及模块功能概述

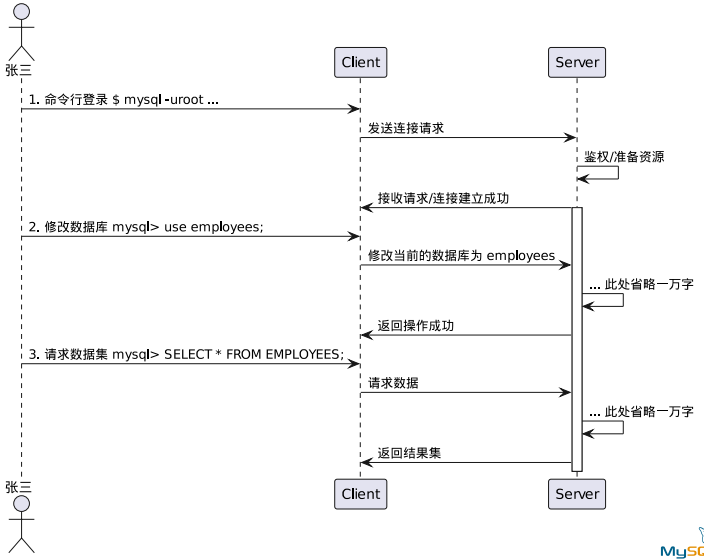
知春路遇上八里桥

<2024-05-10 Fri>

1

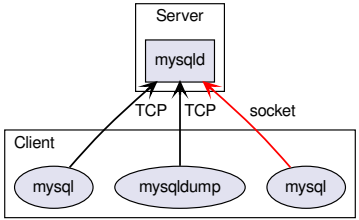
重新认识 MySQL

从日常使用开始


张三
Client
Server
1. 命令行登录 $ mysql -uroot ...
发送连接请求
鉴权/准备资源
接收请求/连接建立成功
2. 修改数据库 mysql> use employees;
修改当前的数据库为 employees
... 此处省略一万字
返回操作成功
3. 请求数据集 mysql> SELECT * FROM EMPLOYEES;
请求数据
... 此处省略一万字
返回结果集
张三
Client
Server

客户端/服务器模型

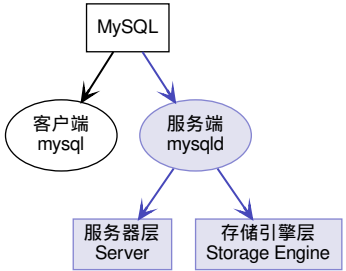
1. 在 MySQL 安装目录的 `$MYSQL_HOME/bin` 中包含一系列程序
2. 客户端程序 (Client) [1]
   - mysql
   - mysqladmin / mysqldump / mysqlcheck
3. 服务器程序 (Server) [2]
   - mysqld
   - mysqld_safe / mysql.server / mysqld_multi



[1] https://dev.mysql.com/doc/refman/8.0/en/programs-client.html
[2] https://dev.mysql.com/doc/refman/8.0/en/programs-server.html

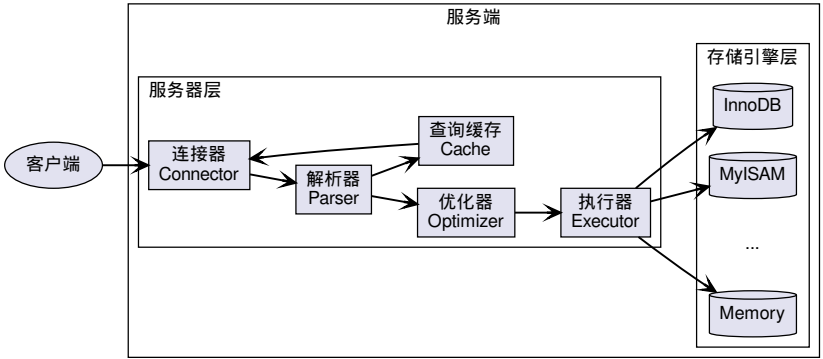
课程的重点部分

1. 服务端部分被分成了以下两个部分：
   - 服务器层：Connector/Parser/Optimizer/Executor 等
   - 存储引擎层：核心分析 InnoDB 存储引擎
2. 服务器层根据功能也被分为多个模块，这个我们后面再进行讨论
3. 相对于服务端，客户端的实现较为简单，一般我们主要讨论服务端设计与实现



简单查询语句执行示意图

1. 针对于一个简单的查询语句，例如：`SELECT * FROM employees;`
2. SQL 执行的基本路径：客户端 → 服务器层 → 存储引擎层



2

服务器层功能

连接器 - 客户端建立连接、获取权限、维持和管理连接

1. MySQL 客户端建连方式
   - Linux 平台下，一种是 TCP 连接，另一种就是 socket 连接
   - Windows 平台下，支持 name pipe 和 share memory 等
2. 连接器会查询权限列表获取该用户的权限
   - 若用户认证通过，则分配线程资源，连接建立成功
   - 修改权限不影响已连接的客户端
   - 只有重新建立连接后才再次鉴权
   - 连接超时通过 wait_timeout 参数控制
3. 查看当前所有连接 `show processlist`

```
mysql> show processlist;
+----+-----------------+-----------+-----------+---------+------+-------------------------+------------------+
| Id | User            | Host      | db        | Command | Time | State                   | Info             |
+----+-----------------+-----------+-----------+---------+------+-------------------------+------------------+
|  5 | event_scheduler | localhost | NULL      | Daemon  |  873 | Waiting on empty queue | NULL             |
|  8 | root            | localhost | employees | Sleep   |  849 |                         | NULL             |
|  9 | root            | localhost | employees | Query   |    0 | init                    | show processlist |
+----+-----------------+-----------+-----------+---------+------+-------------------------+------------------+
3 rows in set, 1 warning (0.01 sec)
```

查询缓存 - 缓存查询结果

1. 连接建立之后，就可以执行查询操作
2. 在一个查询语句中，会先到缓存中查看之前是否查询过这条语句
   - 缓存的 key 是查询的语句，value 是查询的结果
   - 若存在则直接返回缓存的结果；否则继续执行后面的流程
3. 查询缓存功能比较鸡肋，MySQL 8.0 版本已删除了查询缓存功能
   - 通过查询语句作为 key 来说命中率低
   - 缓存失效非常频繁，只要有对一个表的更新，该表所有的查询缓存都会被清空
   - 实际上 MySQL 的性能瓶颈不在缓存
4. query_cache_type 参数控制 `show variables like '%query_cache_type%';`
   - 0(OFF)：关闭，任何情况下都不会使用查询缓存
   - 1(ON)：开启，当语句中使用 SQL_NO_CACHE 提示后，将不使用查询缓存
   - 2(DEMAND)：按需开启，当语句中使用了 SQL_CACHE 提示后，才使用查询缓存

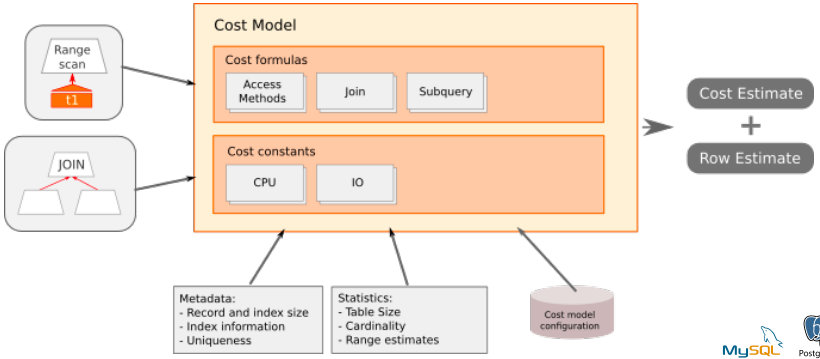
解析器 - 词法分析和语法分析

1. 词法分析
   - 分词操作，将客户端发送的 SQL 语句字符串，解析成 token 流
   - 构建语法树，将 token 流构建成语法树，这里依赖 bison 工具
     `Query_block / Query_expression`
     `Item_xxx / Item_row / Item_cond / Item_result_field / Item_param` 等
2. 语法分析
   - 拿到词法分析的结果，并根据语法规则判断 SQL 语句是否合法。
   - 可能存在语法错误或者语义错误

```
mysql> select * from ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to
your MySQL server version for the right syntax to use near '' at line 1
mysql> select * from aaa;
ERROR 1146 (42S02): Table 'mysql.aaa' doesn't exist
mysql>
```

优化器 - 基于代价模型生成执行计划

1. 解析器通过之后，服务器已经知道了 SQL 语句想要做什么
2. 服务器并不认为我们的 SQL 语句是最优的
3. 为了提高执行效率，它并非完全按照我们的 SQL 语句执行，而要进行一系列优化
   - 表的索引选取
   - Join 的驱动表选择
4. 可以通过 explain 和 OPTIMIZER_TRACE 工具分析 CBO 流程



执行器 - 调用存储引擎接口，返回结果

1. 经过优化器后，接下来就要开始执行了，执行器负责管理执行过程
2. 执行之前，鉴权模块会判断你对该表是否有查询的权限
   - 若有权限则继续执行；
   - 否则会返回如下错误（这里以 SELECT 操作为例，其他操作类似）

   ```
   SELECT command denied to user 'user'@'localhost' for table 't1'
   ```
3. 获取结果，通过接口方式和存储引擎交互，获取数据，然后返回客户端
4. 事务处理，对于 DML 执行器还会进行事务处理，确保数据的安全性和一致性

3

存储引擎层功能

存储引擎

1. 存储引擎（Storage Engine）层主要负责数据的存储和提取
2. 它是直接和磁盘打交道的
3. 以 plugin 的形式存在
4. 常见的存储引擎功能对比 [3]

	MyISAM	InnoDB	Memory
存储限制	256TB	64TB	和主机内存相关
锁	表锁	行锁	表锁
事务	NO	YES	NO
索引	YES	YES	YES
外键	NO	YES	NO

[3] https://dev.mysql.com/doc/refman/8.0/en/storage-engines.html

查看支持的存储引擎

查看存储引擎列表，命令 `show engines`

```
mysql> show engines;
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| Engine             | Support | Comment                                                        | Transactions | XA   | Savepoints |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| ndbcluster         | NO      | Clustered, fault-tolerant tables                               | NULL         | NULL | NULL       |
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary tables      | NO           | NO   | NO         |
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign keys     | YES          | YES  | YES        |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                             | NO           | NO   | NO         |
| MyISAM             | YES     | MyISAM storage engine                                          | NO           | NO   | NO         |
| FEDERATED          | NO      | Federated MySQL storage engine                                 | NULL         | NULL | NULL       |
| ndbinfo            | NO      | MySQL Cluster system information storage engine                | NULL         | NULL | NULL       |
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables                          | NO           | NO   | NO         |
| BLACKHOLE          | YES     | /dev/null storage engine (anything you write to it disappears) | NO           | NO   | NO         |
| CSV                | YES     | CSV storage engine                                             | NO           | NO   | NO         |
| ARCHIVE            | YES     | Archive storage engine                                         | NO           | NO   | NO         |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
11 rows in set (0.00 sec)
```

- Support 表示该引擎是否可用 / Comment 是描述信息
- Transactions 表示是否支持事务 / XA 表示是否支持分布式事务
- Savepoints 表示是否支持回滚

存储引擎操作语句

1. 建表时通过 ENGINE=InnoDB 选项制定存储引擎

```
CREATE TABLE employees (
  -- 此处省略建表语句
) ENGINE=InnoDB;
```

2. 修改表的存储引擎

```
ALTER TABLE employees ENGINE = InnoDB;
```

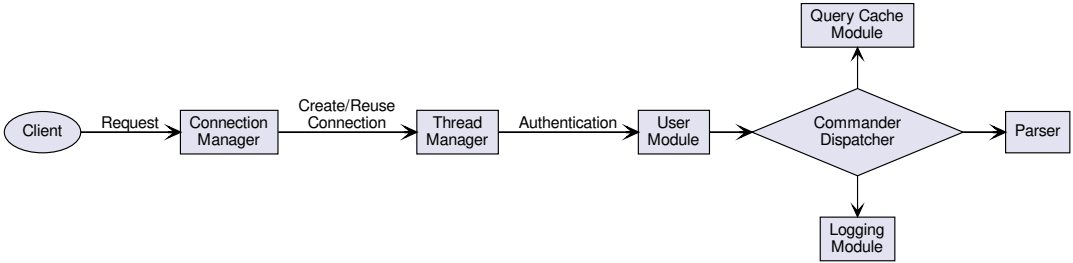
3. 查看表的存储引擎 show create table employees\G

```
mysql> show create table employees\G
*************************** 1. row ***************************
       Table: employees
Create Table: CREATE TABLE `employees` (
  `emp_no` int NOT NULL,
  `birth_date` date NOT NULL,
  `first_name` varchar(14) NOT NULL,
  `last_name` varchar(16) NOT NULL,
  `gender` enum('M','F') NOT NULL,
  `hire_date` date NOT NULL,
  PRIMARY KEY (`emp_no`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
1 row in set (0.01 sec)
```

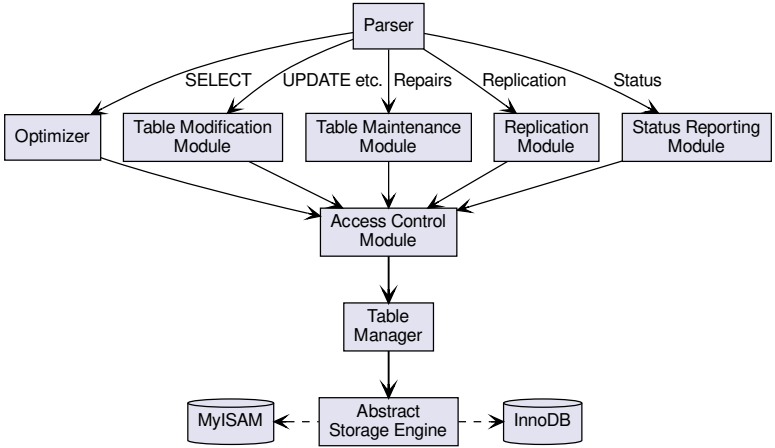
4

真实服务器架构

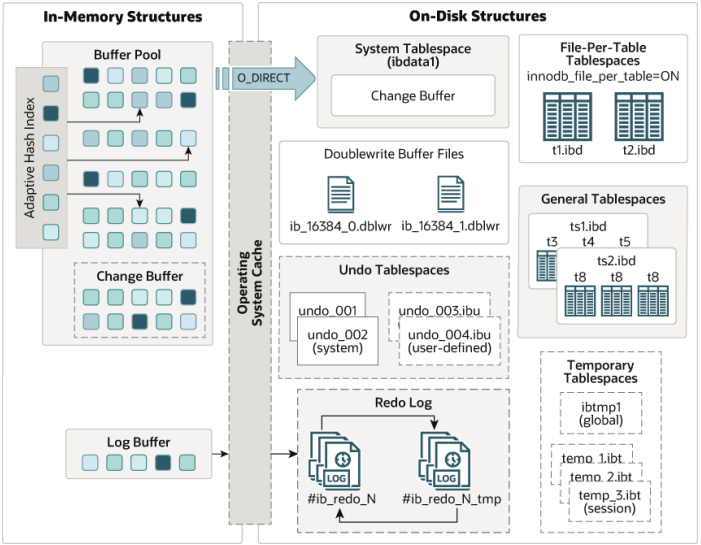
服务端架构：从客户端到解析器



服务端架构：从解析器到存储引擎



InnoDB 引擎 [4]



[4] https://dev.mysql.com/doc/refman/8.0/en/innodb-architecture.html

结束