

# 极客时间 Java 进阶训练营

## 第 12 课

### 性能与SQL优化（1）



KimmKing

Apache Dubbo/ShardingSphere PMC

# 个人介绍

Apache Dubbo/ShardingSphere PMC

前某集团高级技术总监/阿里架构师/某银行北京研发中心负责人

阿里云 MVP、腾讯 TVP、TGO 会员

10 多年研发管理和架构经验

熟悉海量并发低延迟交易系统的设计实现

# 目录

1. 再聊聊性能优化
2. 关系数据库 MySQL
3. 深入数据库原理
4. MySQL 配置优化\*
5. 数据库设计优化\*
6. 第12课总结回顾与作业实践

# 1. 再聊聊性能优化

# 复习一下什么是性能

- 吞吐与延迟：有些结论是反直觉的，指导我们关注什么
- 没有量化就没有改进：监控与度量指标，指导我们怎么去入手
- 80/20原则：先优化性能瓶颈问题，指导我们如何去优化
- 过早的优化是万恶之源：指导我们要选择优化的时机
- 脱离场景谈性能都是耍流氓：指导我们对性能要求要符合实际

性能是一个综合性问题

# DB/SQL 优化是业务系统性能优化的核心

业务系统的分类：计算密集型、数据密集型

业务处理本身无状态，数据状态最终要保存到数据库

一般来说，DB/SQL 操作的消耗在一次处理中占比最大

业务系统发展的不同阶段和时期，性能瓶颈要点不同，类似木桶装水

例如传统软件改成 SaaS 软件

## 2. 关系数据库 MySQL\*

# 什么是关系数据库

- 1970年 Codd 提出关系模型，以关系代数理论为数学基础

《A Relational Model of Data for Large Shared Data Banks》

mathematical sense. Given sets  $S_1, S_2, \dots, S_n$  (not necessarily distinct),  $R$  is a relation on these  $n$  sets if it is a set of  $n$ -tuples each of which has its first element from  $S_1$ , its second element from  $S_2$ , and so on.<sup>1</sup> We shall refer to  $S_j$  as the  $j$ th domain of  $R$ . As defined above,  $R$  is said to have degree  $n$ . Relations of degree 1 are often called unary. degree 2 binary, degree 3 ternary, and degree  $n$   $n$ -ary.

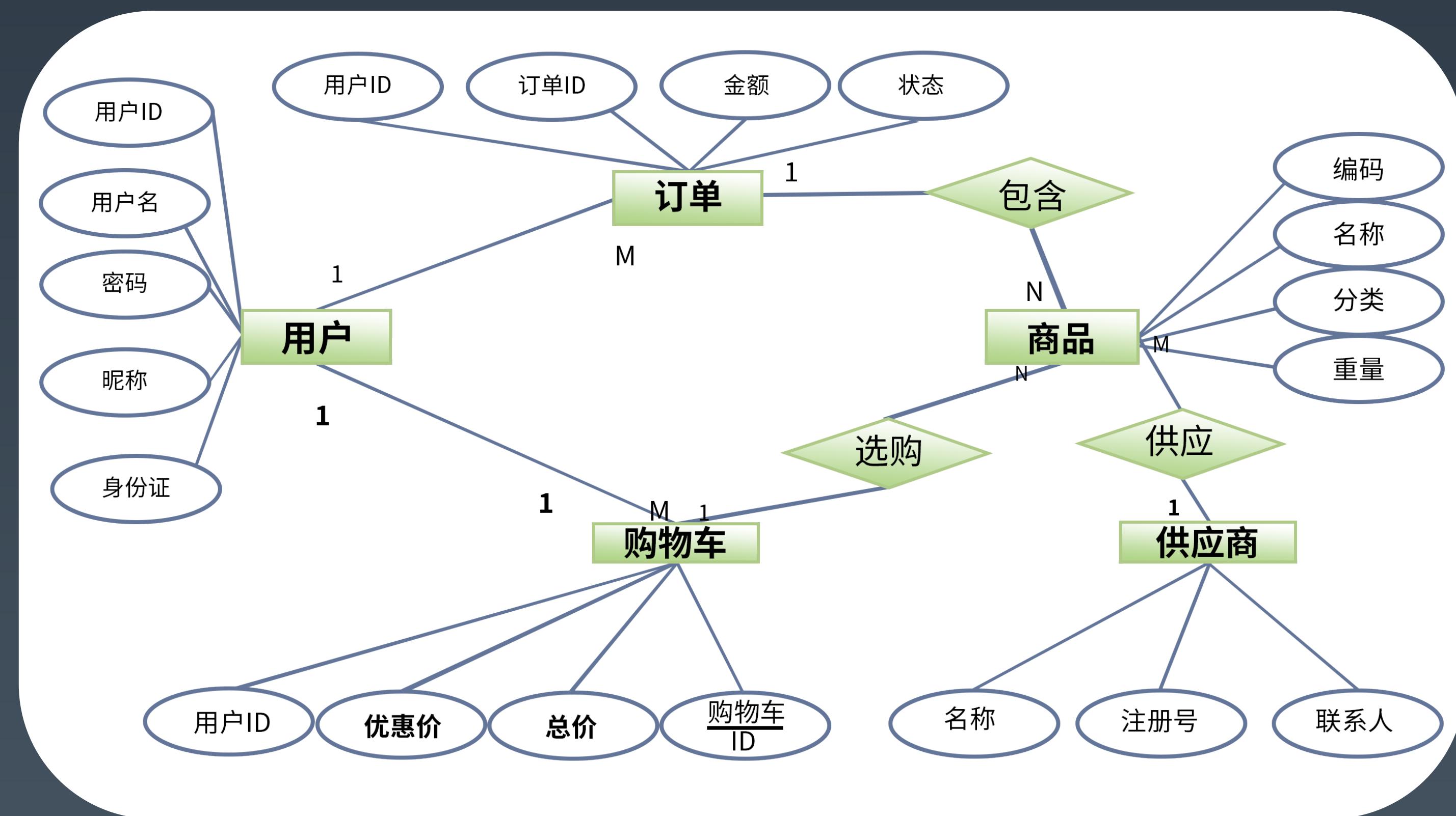
<sup>1</sup> More concisely,  $R$  is a subset of the Cartesian product  $S_1 \times S_2 \times \dots \times S_n$ .

ray which represents an  $n$ -ary relation  $R$  has the following properties:

- (1) Each row represents an  $n$ -tuple of  $R$ .
- (2) The ordering of rows is immaterial.
- (3) All rows are distinct.
- (4) The ordering of columns is significant — it corresponds to the ordering  $S_1, S_2, \dots, S_n$  of the domains on which  $R$  is defined (see, however, remarks below on domain-ordered and domain-unordered relations).
- (5) The significance of each column is partially conveyed by labeling it with the name of the corresponding domain.

# 什么是关系数据库

- E-R 图



# 什么是关系数据库

## 数据库设计范式

第一范式 (1NF)：关系 R 属于第一范式，当且仅当 R 中的每一个属性 A 的值域只包含原子项

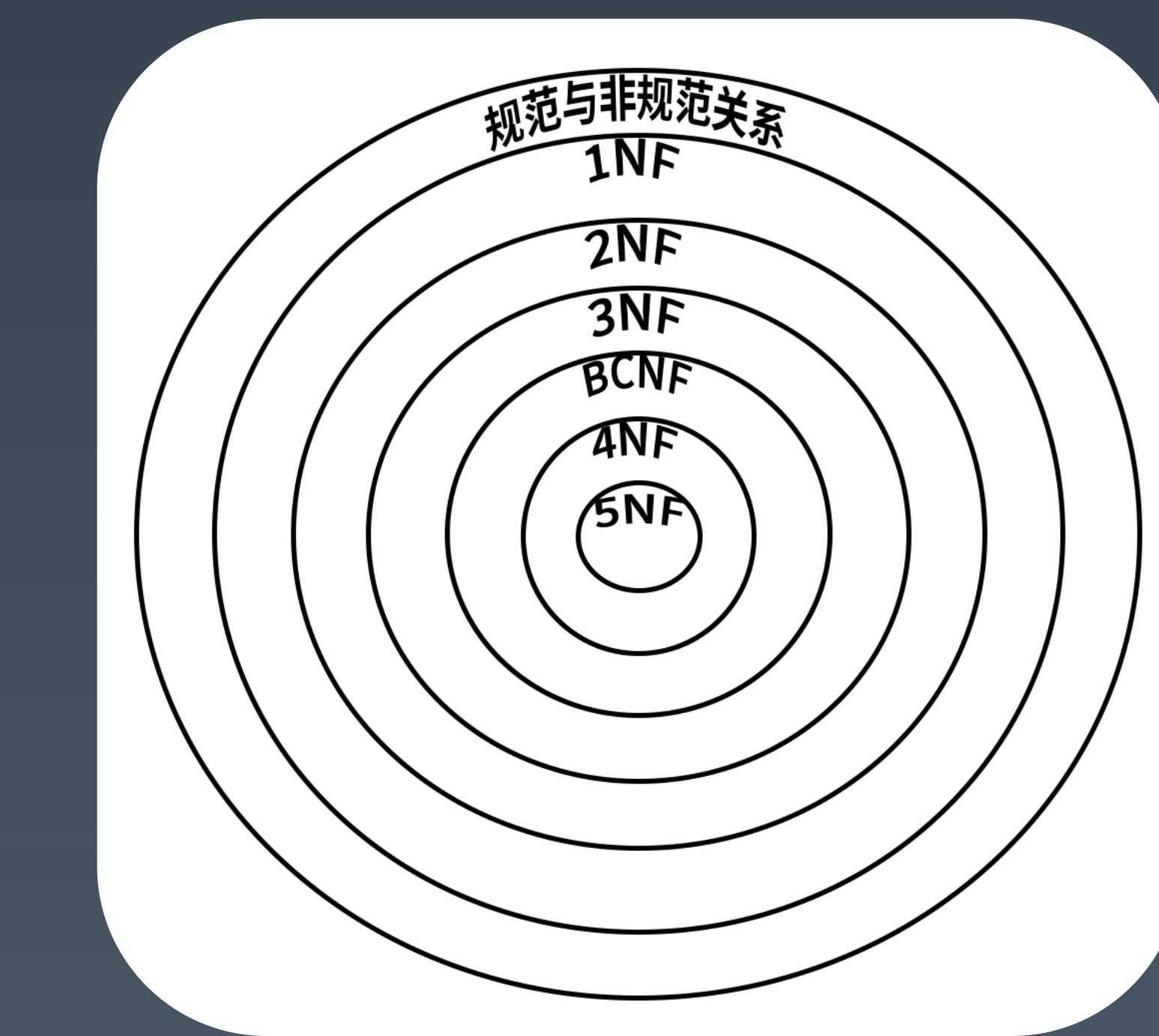
第二范式 (2NF)：在满足 1NF 的基础上，消除非主属性对码的部分函数依赖

第三范式 (3NF)：在满足 2NF 的基础上，消除非主属性对码的传递函数依赖

BC 范式 (BCNF)：在满足 3NF 的基础上，消除主属性对码的部分和传递函数依赖

第四范式 (4NF)：消除非平凡的多值依赖

第五范式 (5NF)：消除一些不合适的连接依赖



# 什么是关系数据库

## 数据库设计范式

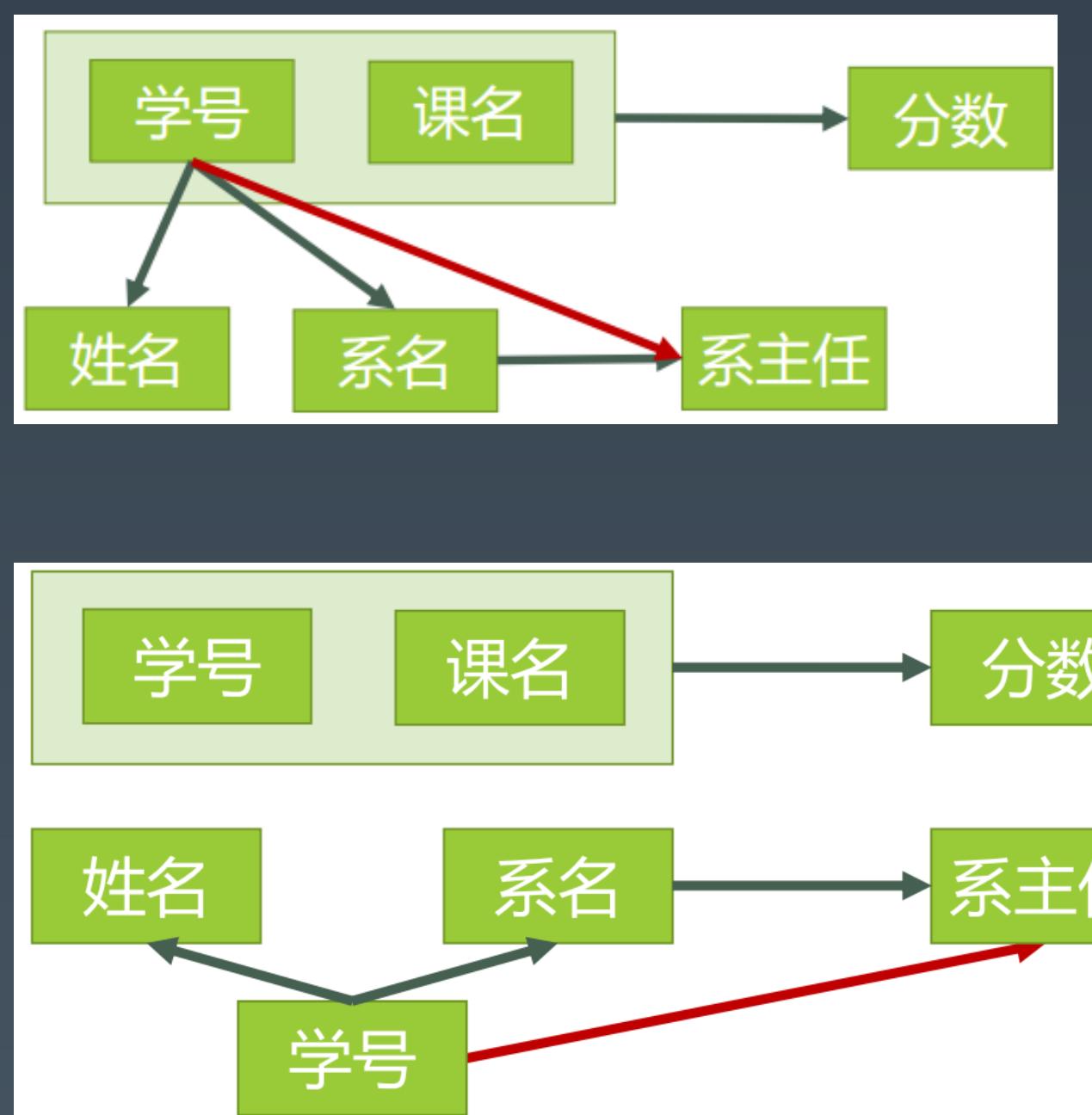
1NF：消除重复数据，即每一列都是不可再分的基本数据项；  
每个列都是原子的。

学号	姓名	系名	系主任	课名	分数
1022211101	李小明	经济系	王强	高等数学	95
1022211101	李小明	经济系	王强	大学英语	87
1022211101	李小明	经济系	王强	普通化学	76
1022211102	张莉莉	经济系	王强	高等数学	72
1022211102	张莉莉	经济系	王强	大学英语	98
1022211102	张莉莉	经济系	王强	计算机基础	88
1022511101	高芳芳	法律系	刘玲	高等数学	82
1022511101	高芳芳	法律系	刘玲	法学基础	82

# 什么是关系数据库

## 数据库设计范式

2NF：消除部分依赖，表中没有列只与主键的部分相关，即每一行都被主键唯一标识；  
每个表都有主键。



The diagram contains two tables within a large rounded rectangle:

学号	课名	分数
1022211101	高等数学	95
1022211101	大学英语	87
1022211101	普通化学	76
1022211102	高等数学	72
1022211102	大学英语	98
1022211102	计算机基础	88
1022511101	高等数学	82
1022511101	法学基础	82

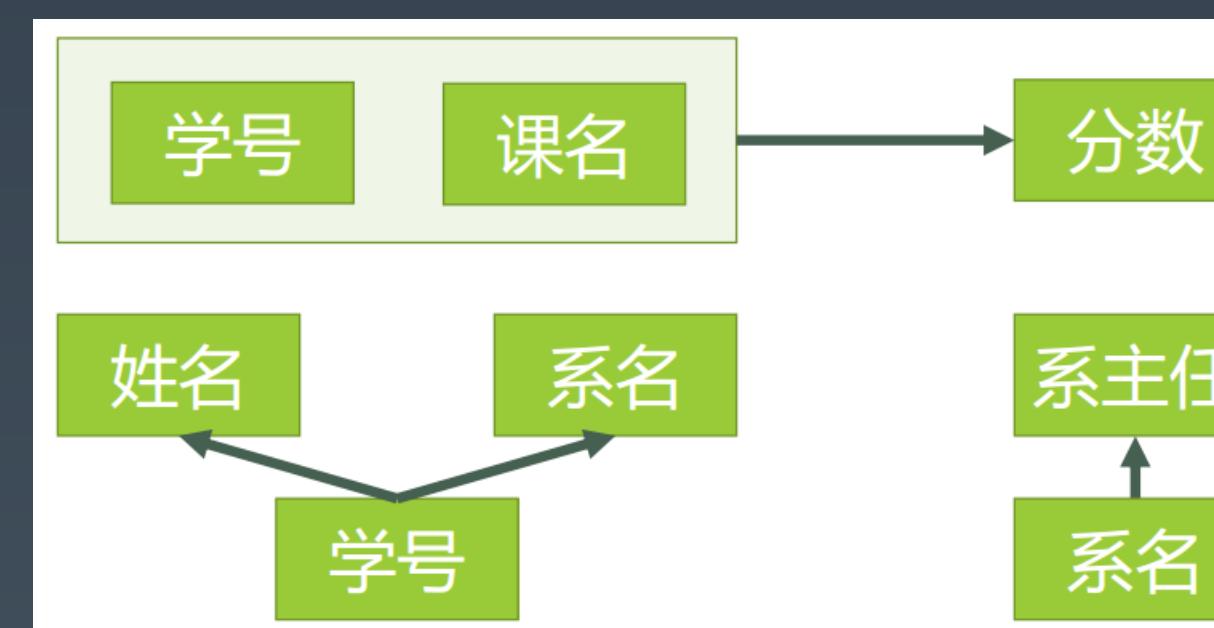
学号	姓名	系名	系主任
1022211101	李小明	经济系	王强
1022211102	张莉莉	经济系	王强
1022511101	高芳芳	法律系	刘玲

# 什么是关系数据库

## 数据库设计范式

3NF：消除传递依赖，消除表中列不依赖主键，而是依赖表中的非主键列的情况，即没有列是与主键不相关的。

从表只引用主表的主键，  
即表中每列都和主键相关。



学号	课名	分数
1022211101	高等数学	95
1022211101	大学英语	87
1022211101	普通化学	76
1022211102	高等数学	72
1022211102	大学英语	98
1022211102	计算机基础	88
1022511101	高等数学	82
1022511101	法学基础	82

学号	姓名	系名
1022211101	李小明	经济系
1022211102	张莉莉	经济系
1022511101	高芳芳	法律系

系名	系主任
经济系	王强
经济系	王强
法律系	刘玲

# 什么是关系数据库

## 数据库设计范式

BCNF: Boyce-Codd Normal Form (巴斯-科德范式)

3NF 的基础上消除主属性对于码的部分与传递函数依赖。

仓库名	管理员	物品名	数量
上海仓	张三	iPhone 5s	30
上海仓	张三	iPad Air	40
北京仓	李四	iPhone 5s	50
北京仓	李四	iPad Mini	60

仓库 (仓库名, 管理员)

库存 (仓库名, 物品名, 数量)

# 常见关系数据库

## 常见关系数据库

开源: MySQL、PostgreSQL

商业: Oracle, DB2, SQL Server

内存数据库: Redis? , VoltDB

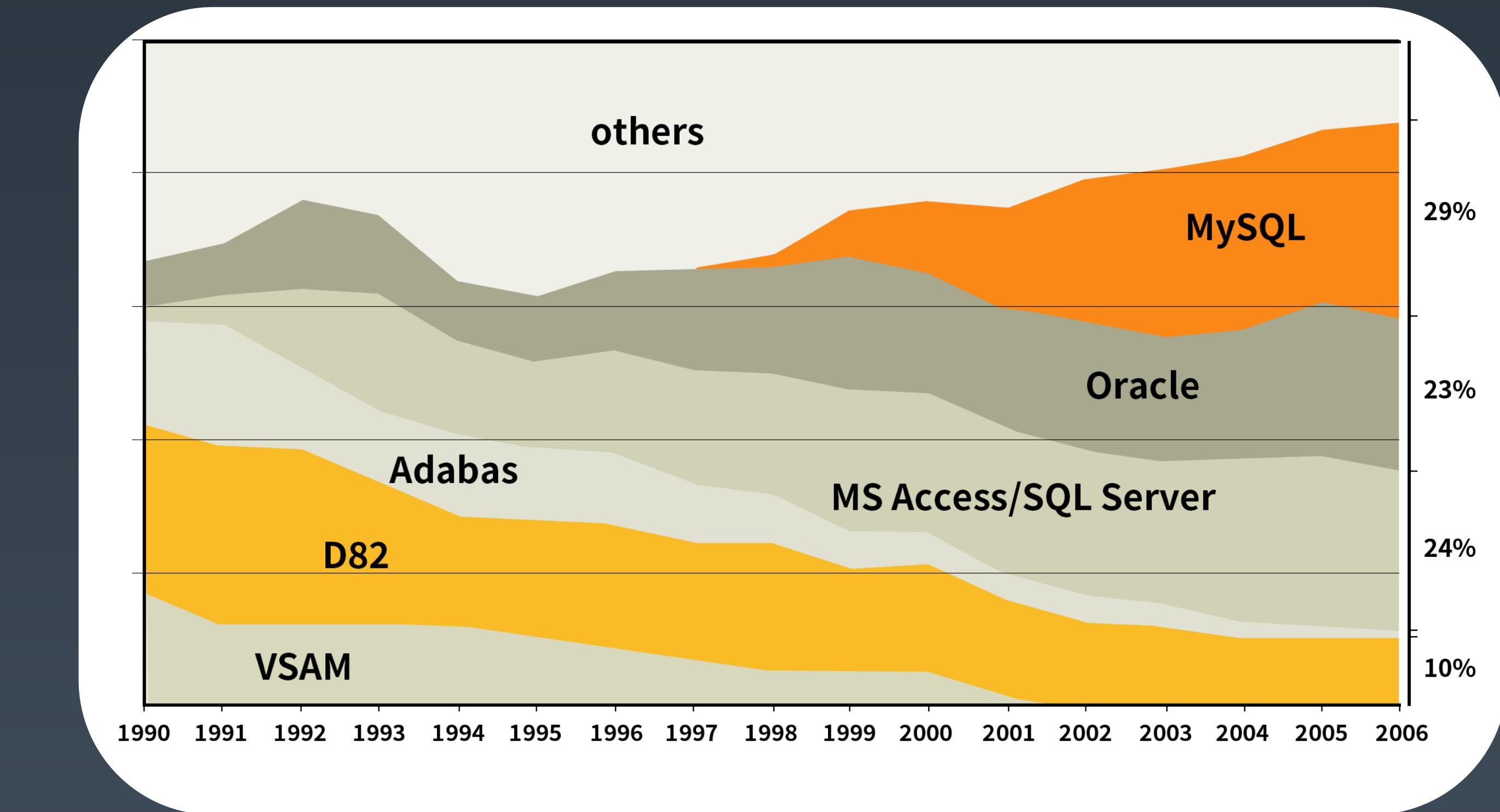
图数据库: Neo4j, Nebula

时序数据库: InfluxDB、openTSDB

其他关系数据库: Access、Sqlite、H2、Derby、Sybase、Infomix 等

NoSQL 数据库: MongoDB、Hbase、Cassandra、CouchDB

NewSQL/分布式数据库: TiDB、CockroachDB、NuoDB、OpenGauss、OB、TDSQL



# SQL 语言

SQL 语言1974年由 Boyce 和 Chamberlin 提出，并首先在 IBM 公司研制的关系数据库系统 SystemR 上实现。

1979年 ORACLE 公司首先提供商用的 SQL，IBM 公司在 DB2 和 SQL/DS 数据库系统中也实现了 SQL。

1986年10月，美国 ANSI 采用 SQL 作为关系数据库管理系统的标准语言（ANSI X3.135-1986），后为国际标准化组织（ISO）采纳为国际标准。

1989年，美国 ANSI 采纳在 ANSI X3.135-1989报告中定义的关系数据库管理系统的 SQL 标准语言，称为 ANSI SQL 89，该标准替代 ANSI X3.135-1986版本。

思考：SQL 语言是不是操作数据库必须的？

# SQL 语言

结构化查询语言包含6个部分：

- 1、数据查询语言（**DQL**: Data Query Language）：其语句，也称为“数据检索语句”，用以从表中获得数据，确定数据怎样在应用程序给出。保留字 SELECT 是 DQL (也是所有 SQL) 用得最多的动词，其他 DQL 常用的保留字有 WHERE, ORDER BY, GROUP BY 和 HAVING。这些 DQL 保留字常与其它类型的 SQL 语句一起使用。
- 2、数据操作语言（**DML**: Data Manipulation Language）：其语句包括动词 INSERT、UPDATE 和 DELETE。它们分别用于添加、修改和删除。
- 3、事务控制语言（**TCL**）：它的语句能确保被 DML 语句影响的表的所有行及时得以更新。包括 COMMIT (提交) 命令、SAVEPOINT (保存点) 命令、ROLLBACK (回滚) 命令。
- 4、数据控制语言（**DCL**）：它的语句通过 GRANT 或 REVOKE 实现权限控制，确定单个用户和用户组对数据库对象的访问。某些 RDBMS 可用 GRANT 或 REVOKE 控制对表单个列的访问。
- 5、数据定义语言（**DDL**）：其语句包括动词 CREATE,ALTER 和 DROP。在数据库中创建新表或修改、删除表 (CREATE TABLE 或 DROP TABLE)；为表加入索引等。
- 6、指针控制语言（**CCL**）：它的语句，像 DECLARE CURSOR, FETCH INTO 和 UPDATE WHERE CURRENT 用于对一个或多个表单独行的操作。

# SQL 语言

## SQL 的各个版本：

1986年, ANSI X3.135-1986, ISO/IEC 9075:1986, SQL-86

1989年, ANSI X3.135-1989, ISO/IEC 9075:1989, SQL-89

1992年, ANSI X3.135-1992, ISO/IEC 9075:1992, SQL-92 (SQL2)

1999年, ISO/IEC 9075:1999, SQL:1999 (SQL3)

2003年, ISO/IEC 9075:2003, SQL:2003

2008年, ISO/IEC 9075:2008, SQL:2008

2011年, ISO/IEC 9075:2011, SQL:2011

# MySQL 数据库

瑞典的 MySQL AB 创立于1995年。

2008年1月16日 MySQL AB 被 Sun Microsystems 收购。

2009年4月20日，甲骨文（Oracle）收购 Sun Microsystems 公司。

其后分离成两个版本：MariaDB 和 MySQL



# MySQL 数据库

## MySQL 的版本

- 4.0 支持 InnoDB，事务
- 2003年，5.0
- 5.6 ==> 历史使用最多的版本
- 5.7 ==> 近期使用最多的版本
- 8.0 ==> 最新和功能完善的版本

选择学习哪个版本？

# MySQL 数据库

## - 5.6/5.7的差异

5.7支持：

- 多主
- MGR 高可用
- 分区表
- json
- 性能
- 修复 XA 等

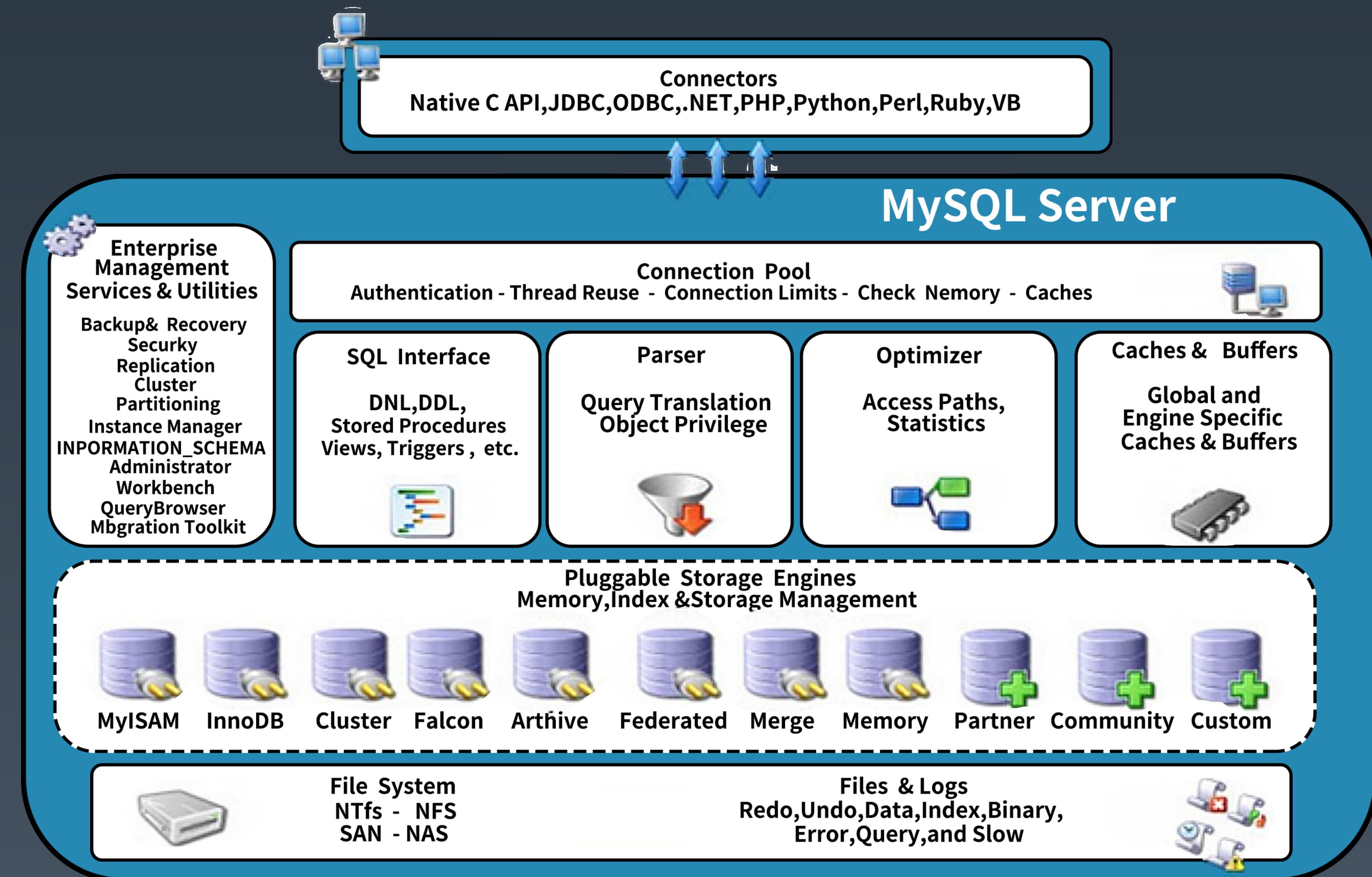
# MySQL 数据库

## - 5.7/8.0的差异

- 通用表达式
- 窗口函数
- 持久化参数
- 自增列持久化
- 默认编码 utf8mb4
- DDL 原子性
- JSON 增强
- 不再对 group by 进行隐式排序? ? ==> 坑

### 3. 深入数据库原理\*

# MySQL 架构图



# MySQL 存储

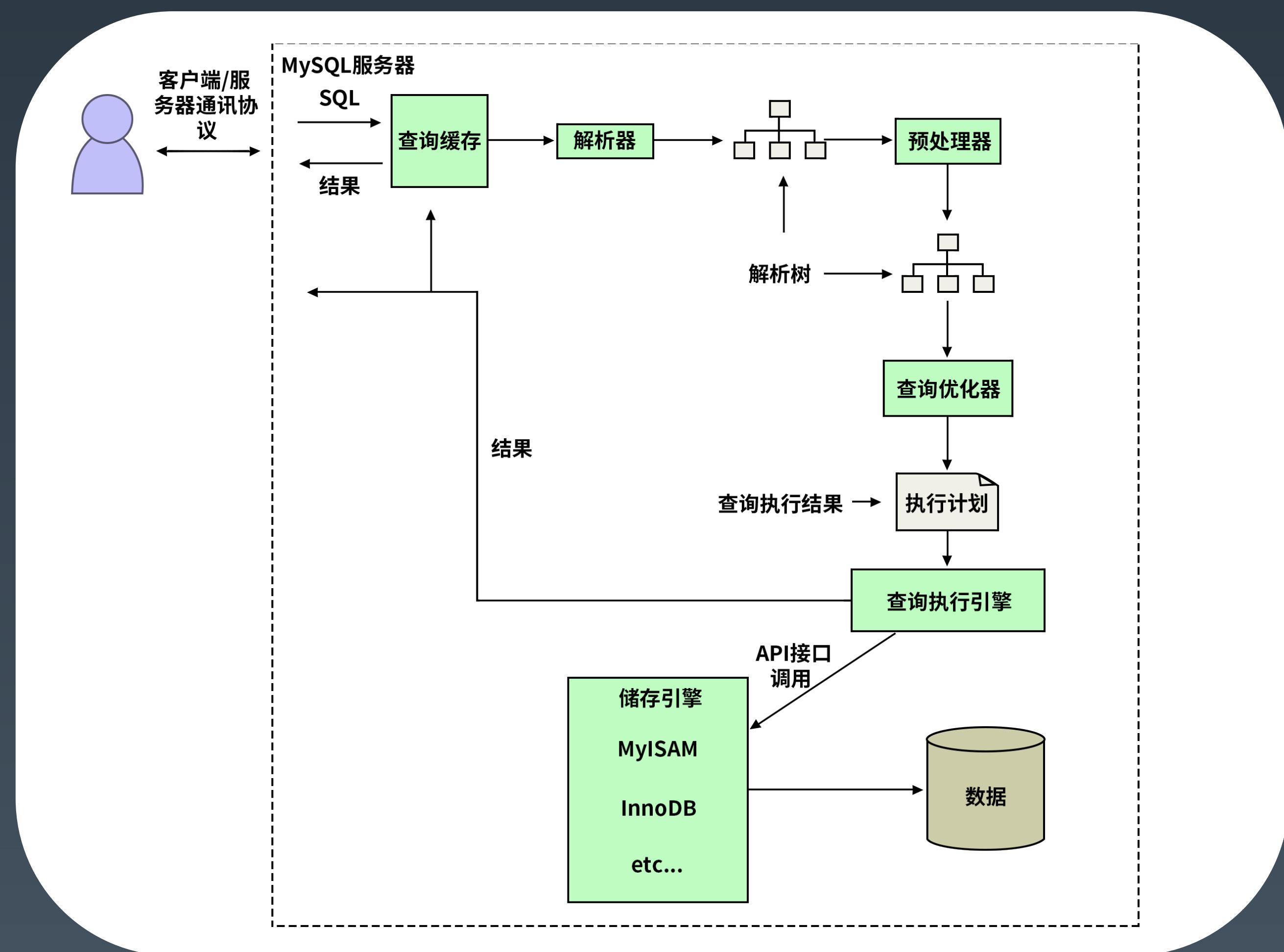
## 独占模式

- 1) 、日志组文件: ib\_logfile0 和 ib\_logfile1, 默认均为5M
- 2) 、表结构文件: \*.frm
- 3) 、独占表空间文件: \*.ibd
- 4) 、字符集和排序规则文件: db.opt
- 5) 、binlog 二进制日志文件: 记录主数据库服务器的 DDL 和 DML 操作
- 6) 、二进制日志索引文件: master-bin.index

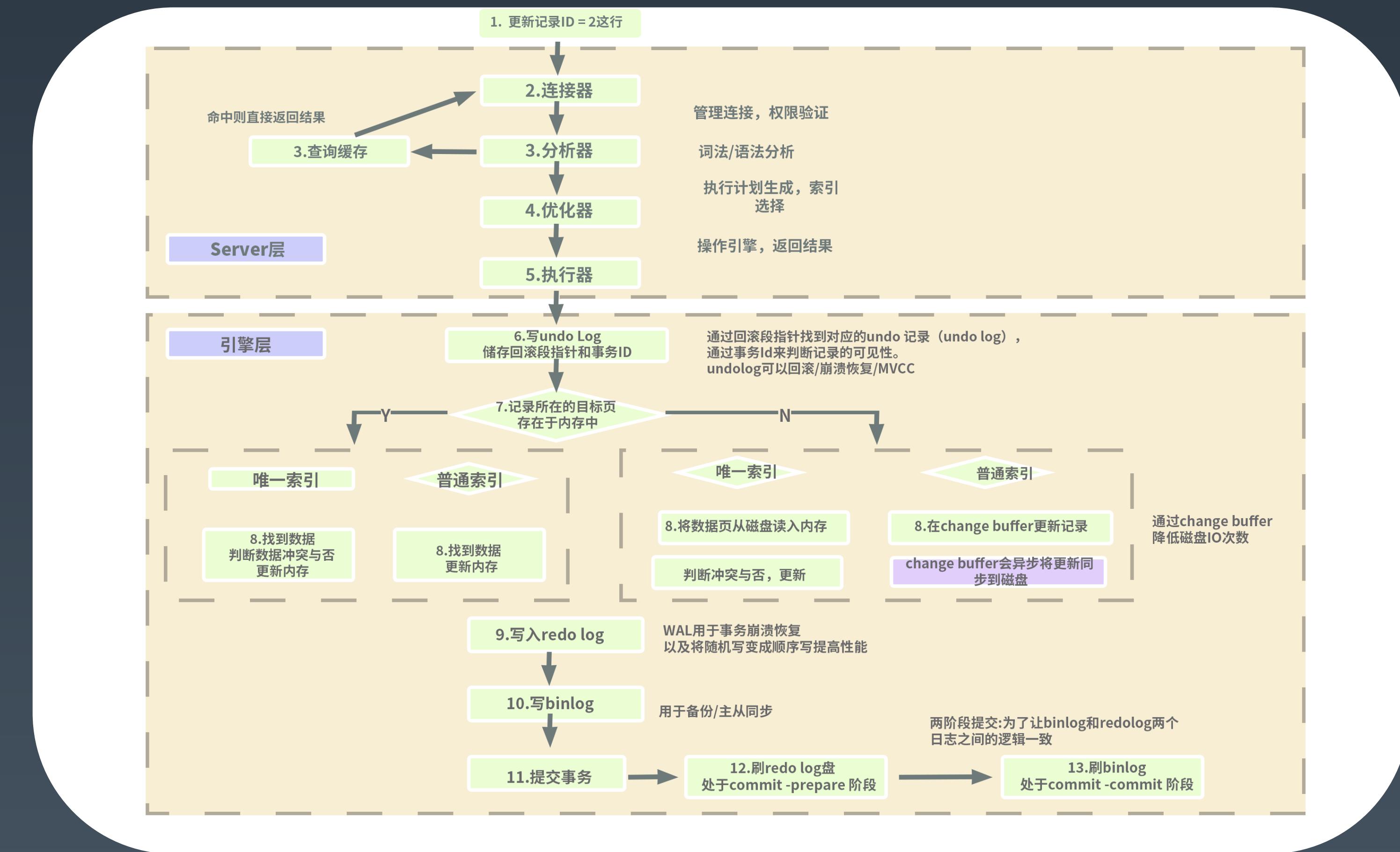
## 共享模式 innodb\_file\_per\_table=OFF

- 1) 、数据都在 ibdata1

# MySQL 简化执行流程



# MySQL 详细执行流程



# MySQL 执行引擎和状态

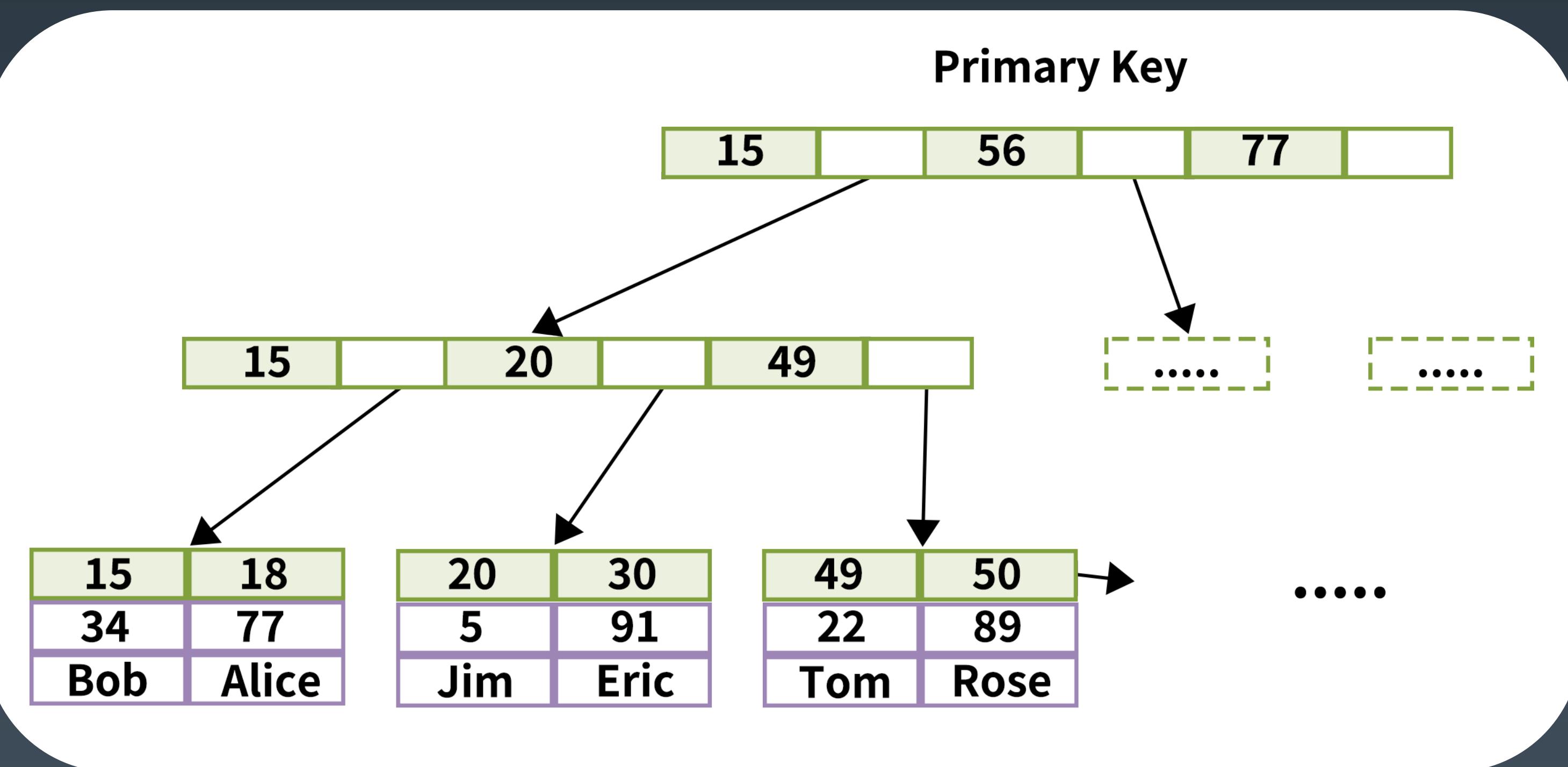
存储引擎	myisam	innodb	memory	archive
存储限制	256TB	64TB	有	无
事务	-	支持	-	-
索引	支持	支持	支持	-
锁的粒度	表锁	行锁	表锁	行锁
数据压缩	支持	-	-	支持
外键	-	支持	-	-

# MySQL 对 SQL 执行顺序



实际上这个过程也并不是绝对这样的，中间 mysql 会有部分的优化以达到最佳的优化效果，比如在 select 筛选出找到的数据集

# MySQL 索引原理



数据是按页来分块的，当一个数据被用到时，其附近的数据也通常会马上被使用。

InnoDB 使用 B+ 树实现聚集索引，

为什么一般单表数据不超过2000万？

# MySQL 数据库操作演示

## 操作示例

- 安装的几种方式，安装文件或命令，docker
- 操作工具，mysql-cli 或 IDE(DataGrip,MySQL-WorkBench,MySQL-Front,Navicat 等)
- MySQL 库结构，操作语句与命令
- MySQL SQL 语法演示

5.6/5.7/8.0

# 4. MySQL 配置优化\*

# 参数配置优化

## 查看参数配置

- show variables like xxx

my.cnf 文件 // my.ini

[mysqld]

server

[mysql]

client

# 参数配置优化-1

## 1) 连接请求的变量

- 1、max\_connections
- 2、back\_log
- 3、wait\_timeout和interative\_timeout

# 参数配置优化-2

## 2) 缓冲区变量

- 4、key\_buffer\_size
- 5、query\_cache\_size (查询缓存简称 QC)
- 6、max\_connect\_errors
- 7、sort\_buffer\_size
- 8、max\_allowed\_packet=32M
- 9、join\_buffer\_size=2M
- 10、thread\_cache\_size=300

# 参数配置优化-3

## 3) 配置 Innodb 的几个变量

- 11、innodb\_buffer\_pool\_size=128M
- 12、innodb\_flush\_log\_at\_trx\_commit
- 13、innodb\_thread\_concurrency=0
- 14、innodb\_log\_buffer\_size
- 15、innodb\_log\_file\_size=50M
- 16、innodb\_log\_files\_in\_group=3
- 17、read\_buffer\_size=1M
- 18、read\_rnd\_buffer\_size=16M
- 19、bulk\_insert\_buffer\_size=64M
- 20、binary log

# 5.数据库设计优化\*

# MySQL 数据库设计优化-最佳实践

- 如何恰当选择引擎?
- 库表如何命名?
- 如何合理拆分宽表?
- 如何选择恰当数据类型: 明确、尽量小
  - char、varchar 的选择
  - (text/blob/clob) 的使用问题?
  - 文件、图片是否要存入到数据库?
  - 时间日期的存储问题?
  - 数值的精度问题?
  - 是否使用外键、触发器?

# MySQL 数据库设计优化-最佳实践

- 唯一约束和索引的关系？
- 是否可以冗余字段？
- 是否使用游标、变量、视图、自定义函数、存储过程？
- 自增主键的使用问题？
- 能够在线修改表结构（DDL 操作）？
- 逻辑删除还是物理删除？
- 要不要加 `create_time,update_time` 时间戳？
- 数据库碎片问题？
- 如何快速导入导出、备份数据？

性能是一个综合性问题

# 6. 总结回顾与作业实践

# 第 12 节课总结回顾

性能与关系数据库

MySQL 与 SQL

数据库原理

参数优化与设计优化

# 第12节课作业实践

- 1、（选做）：基于课程中的设计原则和最佳实践，分析是否可以将自己负责的业务系统进行数据库设计或是数据库服务器方面的优化。
- 2、（必做）：基于电商交易场景（用户、商品、订单），设计一套简单的表结构，提交 DDL 的 SQL 文件到 Github（后面2周的作业依然要是用到这个表结构）。
- 3、（选做）：尽可能多的从“常见关系数据库”中列的清单，安装运行，并使用上一题的 SQL 测试简单的增删改查。
- 4、（选做）：基于上一题，尝试对各个数据库测试100万订单数据的增删改查性能。
- 5、（选做）：尝试对 MySQL 不同版本/引擎下测试100万订单数据的增删改查性能。
- 6、（选做）：模拟1000万订单数据，测试不同方式下导入导出（数据备份还原）MySQL 的速度，包括 jdbc 程序处理和命令行处理。思考和实践，如何提升处理效率。
- 7、（选做）：对 MySQL 配置不同的数据库连接池（DBCP、C3P0、Druid、Hikari），测试增删改查100万次，对比性能，生成报告。