

# MySQL 常见操作与原理分析

公益讲座11:00准时开始,请大家先浏览云技术微信公众号技术文章。资料会在各群同步发布,已入群客户请勿重复入群!



20-21

数据库和云讲座群



甲骨文云技术公众号



B站专家系列课程



# 基于 Oracle 数据库 免费企业数据健康检查

- 及时了解数据库健康状况，发现并解决潜在问题
- 维护数据库系统良好状态，保护数据资产的安全
- 提升数据库性能、稳定性和安全性，降低业务风险

免费咨询热线：

**400-699-8888**

\* 活动最终解释权归甲骨文公司所有

# MySQL常见操作与原理分析

甲骨文技术公益课 - 数据库专场

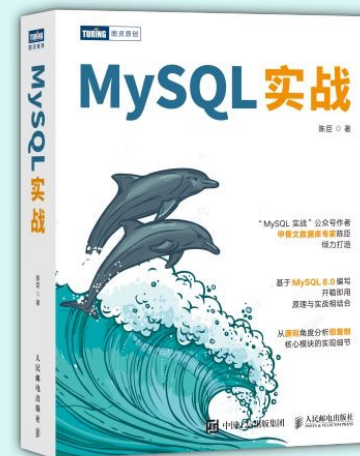
2023年7月7日 11:00

线上直播

## 个人简介

- 陈臣
- 《MySQL实战》作者。
- 公众号“MySQL实战”作者。
- 博客园推荐博客博主iVictor，博客全网阅读量超 300 万。
- 超过 10 年的数据库管理和架构经验。
- 擅长 MySQL 数据库日常操作的原理剖析。

# Oracle数据库专家 5年磨一剑



基于  
**MySQL 8.0**

.....

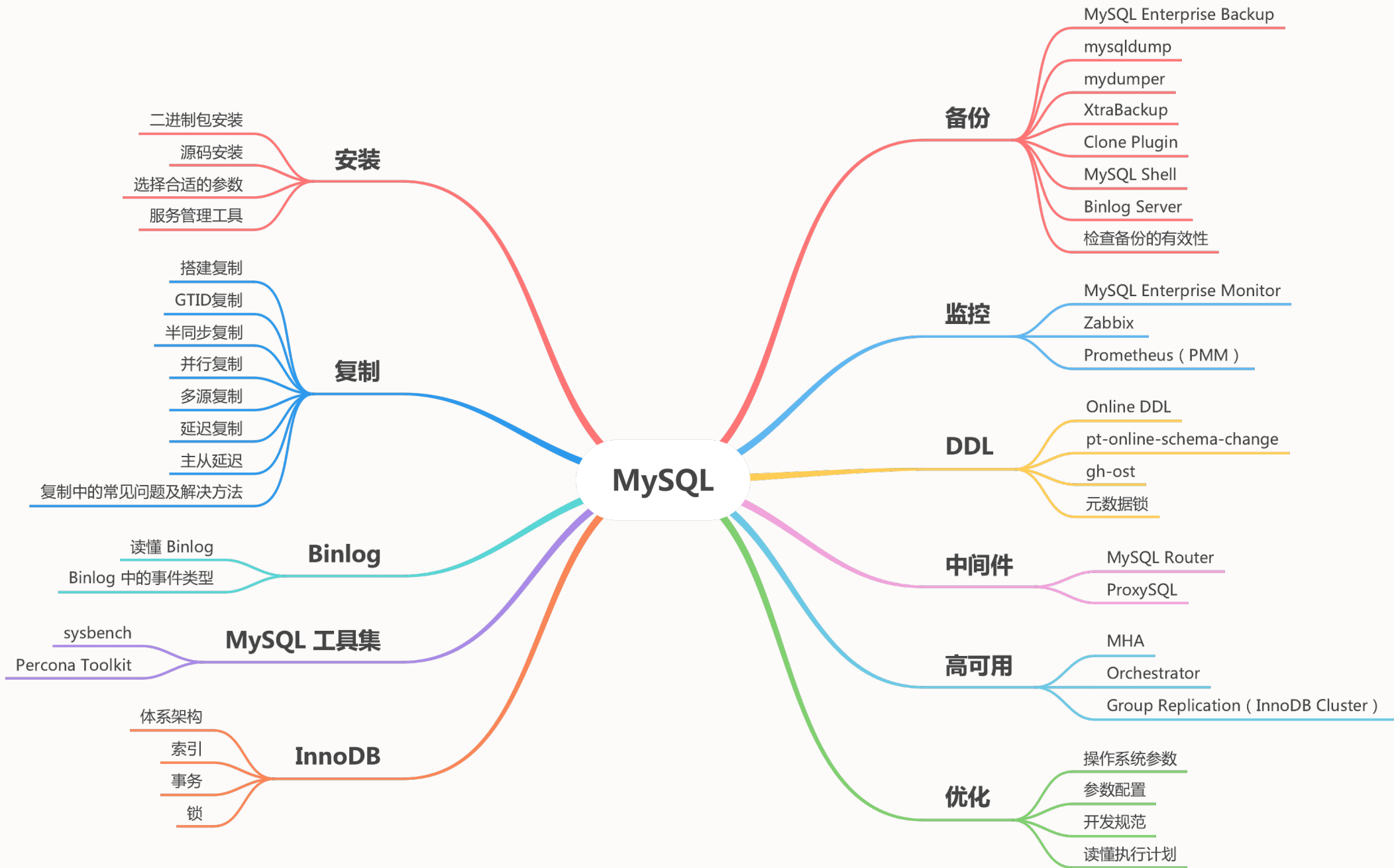
**内容全面  
实战和原理相结合**

# 议程

- 1 一图读懂 MySQL 知识体系
- 2 MySQL 常见的高可用方案
- 3 MySQL 常见操作与原理分析
- 4 MySQL 8.0 的新特性
- 5 MySQL 学习资料推荐



# 一图读懂 MySQL 知识体系



# MySQL 常见的高可用方案

---



# MySQL 常见的高可用方案

## 双主 + Keepalived

- 优点
  - 简单
- 缺点
  - 很难保证数据的一致性
  - 不适用于一主多从的场景

## MHA

- 优点
  - 能做到在 0~30s 内自动完成数据库的故障切换操作，在 master 服务器不宕机的情况下，基本能保证数据的一致性。
  - 对业务基本无侵入
  - 能自动补偿 Binlog
- 缺点
  - 不适用于复杂的集群拓扑
  - 需要做 SSH 互信
  - 使用 Perl 开发，二次开发困难
  - 多年不维护，不适用于 MySQL 8.0

## Orchestrator

- 优点
  - 多点探测
  - 使用 Raft 实现 Orchestrator 的高可用
  - 自动发现集群拓扑
  - 可通过界面拖拽管理集群拓扑
  - 有丰富的 API 接口
  - 定义了多个 hook，可自定义脚本
- 缺点
  - 无 Binlog 补偿机制

## MGR

- 优点
  - 官方原生的高可用解决方案
  - 基于 Paxos 协议，能保证数据的强一致性
  - InnoDB Cluster，降低了 MGR 的管理和维护成本
- 缺点
  - 至少需三节点



# MySQL 常见操作与原理分析

---

## 没做读写分离，为什么也要关注主从延迟

主从延迟带来的问题：

- 对于读写分离的业务，主从延迟意味着业务会读到旧数据。
- 主从延迟过大，会影响数据库的高可用切换。这一点尤其需要注意。
  1. 如果等待从库应用完差异的 Binlog 才做高可用切换，无疑会影响数据库服务的可用性。
  2. 如果不等待，直接切换，则意味着没应用完的这部分 Binlog 的数据会丢失，业务不一定能接受这种情况。



# MySQL 主从延迟的常见原因及解决方法

## 主从延迟的常见原因及解决方法

### IO 线程存在延迟

- 网络延迟 可开启 `slave_compressed_protocol` 参数, 启用压缩传输二进制日志
- 磁盘 IO 存在瓶颈
  - 可调整从库的双一设置
  - 关闭 binlog
- 网卡存在问题 可通过 scp 远程拷贝文件测试网络的带宽

### SQL 线程存在延迟

- SQL 线程单线程重放 开启并行复制
- STATEMENT 格式下的慢 SQL 开启 `log_slow_slave_statements`, 优化慢 SQL
- ROW 格式
  - 表上没有任何索引 添加索引
  - 将 `slave_rows_search_algorithms` 设置为 `INDEX_SCAN,HASH_SCAN`
  - 大事务 分批执行
- 从库上有查询操作
  - 查询, 占用了大量系统资源
  - 锁等待, 从库的查询操作堵塞了主库的 DDL 操作
- 备份 全局读锁堵塞了 SQL 线程的重放操作
- 磁盘 IO 存在瓶颈
  - 可调整从库的双一设置
  - 关闭 binlog



# 线上对大表加字段，会不会锁表

## pt-online-schema-change的实现原理

### 1. 获取原表的表结构

```
SHOW CREATE TABLE `slowtech`.`t1`
```

### 2. 创建一张新表，表结构同原表一致

```
CREATE TABLE `slowtech`.`t1_new` (  
  `id` int NOT NULL AUTO_INCREMENT,  
  `name` varchar(10) DEFAULT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=9633541
```

### 3. 对新表进行 DDL

```
ALTER TABLE `slowtech`.`t1_new` add column c1 datetime
```

### 4. 创建触发器

DELETE

```
CREATE TRIGGER `pt_osc_slowtech_t1_del` AFTER DELETE ON  
`slowtech`.`t1` FOR EACH ROW DELETE IGNORE FROM `slowtech`.`t1_new`  
WHERE `slowtech`.`t1_new`.`id` <=> OLD.`id`
```

UPDATE

```
CREATE TRIGGER `pt_osc_slowtech_t1_upd` AFTER UPDATE ON  
`slowtech`.`t1` FOR EACH ROW BEGIN DELETE IGNORE FROM  
`slowtech`.`t1_new` WHERE !(OLD.`id` <=> NEW.`id`) AND  
`slowtech`.`t1_new`.`id` <=> OLD.`id`;REPLACE INTO  
`slowtech`.`t1_new` (`id`,`name`) VALUES (NEW.`id`, NEW.`name`);END
```

INSERT

```
CREATE TRIGGER `pt_osc_slowtech_t1_ins` AFTER INSERT ON  
`slowtech`.`t1` FOR EACH ROW REPLACE INTO `slowtech`.`t1_new` (`id`,  
`name`) VALUES (NEW.`id`, NEW.`name`)
```

### 5. 以 chunk 为单位拷贝数据

```
INSERT LOW_PRIORITY IGNORE INTO `slowtech`.`t1_new` (`id`,`name`)  
SELECT `id`,`name` FROM `slowtech`.`t1` FORCE INDEX(PRIMARY)  
WHERE ((`id` >= '1') AND ((`id` <= '10501')) LOCK IN SHARE MODE /*pt-  
online-schema-change 15935 copy nibble*/
```

共享锁，会堵塞客户端对这些记录的 DML 操作。

### 6. 拷贝完数据，会对新表执行 ANALYZE 操作

```
ANALYZE TABLE `slowtech`.`t1_new`
```

### 7. 对原表和新表进行 RENAME 操作

```
RENAME TABLE `slowtech`.`t1` TO `slowtech`.`t1_old`,  
`slowtech`.`t1_new` TO `slowtech`.`t1`
```

### 8. 删除 RENAME 前的原表

```
DROP TABLE IF EXISTS `slowtech`.`t1_old`
```

### 9. 删除触发器

DELETE

```
DROP TRIGGER IF EXISTS `slowtech`.`pt_osc_slowtech_t1_del`
```

UPDATE

```
DROP TRIGGER IF EXISTS `slowtech`.`pt_osc_slowtech_t1_upd`
```

INSERT

```
DROP TRIGGER IF EXISTS `slowtech`.`pt_osc_slowtech_t1_ins`
```



# 条件没有索引，怎么批量删除（更新）一批数据

## 两种思路

基于条件从 SLAVE 获取主键值，然后基于主键+条件在主库进行批量操作

基于主键循环遍历

```
select min(id),max(id) from t1;
```

```
for (i=min_id;i<=max_id;i=i+1000) {  
    delete from t1 where create_time < '20221001' and id>=i and id<=i+1000;  
}
```

# MySQL 8.0 的新特性

---

# MySQL 发布史

版本	GA时间	最新的小版本	最新小版本的发布时间	产品支持的结束时间
MySQL 5.1	2008-11-14	5.1.73	2013-12-03	2013-12
MySQL 5.5	2010-12-03	5.5.62	2018-10-22	2018-12
MySQL 5.6	2013-02-05	5.6.51	2021-01-20	2021-02
MySQL 5.7	2015-10-21	5.7.42	2023-04-18	2023-10
MySQL 8.0	2018-04-19	8.0.33	2023-04-18	2026-04





# 开发

- 不可见索引 ( Invisible index )
- 降序索引
- 公用表表达式 ( Common table expressions )
- 窗口函数 ( Window functions )
- 快速加列 ( ALGORITHM=INSTANT )
- 通过 CHECK 设置约束
- 默认字符集由 latin1 更改为 utf8mb4
- Atomic DDL
- HASH JOIN



# 开发

- SELECT ... FOR SHARE 和 SELECT ... FOR UPDATE 语句中引入 NOWAIT 和 SKIP LOCKED 选项，解决电商场景热点行问题
- 正则表达式的增强，新增了4个相关函数，REGEXP\_INSTR(), REGEXP\_LIKE(), REGEXP\_REPLACE(), REGEXP\_SUBSTR()
- BLOB, TEXT, GEOMETRY 和 JSON 字段允许设置默认值
- 自增主键的持久化
- information\_schema 中的 innodb\_locks 和 innodb\_lock\_waits 表被移除，取而代之的是 performance\_schema 中的 data\_locks 和 data\_lock\_waits 表
- JSON字段的更新 ( JSON Partial Updates )

## 直方图

## [MySQL 8.0 21 个开发相关的新特性](#)



# 管理

- 可持久化全局变量 ( SET PERSIST )
- Grant 不再隐式创建用户
- 可通过 RESTART 命令重启 MySQL 实例
- 资源组 ( Resource Groups ) ， 用来控制线程的优先级及其能使用的资源， 目前， 能被管理的资源只有 CPU
- 引入了 innodb\_dedicated\_server 选项， 可基于服务器的内存来动态设置 innodb\_buffer\_pool\_size , innodb\_log\_file\_size 和 innodb\_flush\_method
- 移除 PASSWORD() 函数。 这就意味着无法通过 SET PASSWORD ... = PASSWORD('a uth\_string') 命令修改用户密码

• Clone Plugin

[MySQL 8.0 18个管理相关的新特性](#)



# 安全

- Roles。Role是一组权限的组合。如果一个实例中的账号比较多，通过Role可以简化DBA的日常管理。
- 默认的认证插件由 `mysql_native_password` 更改为 `caching_sha2_password`
- 引入了更多细粒度的权限来替代 SUPER 权限，现在授予 SUPER 权限会提示 warning
- MySQL 5.7 引入的表空间加密特性可对Redo Log、Undo Log、Binlog进行加密
- 可剔除某些 Schema 的授权
- 密码管理：可设置密码的过期时间；可设置密码的复用策略；修改当前用户密码时需指定之前的密码；一个账户可同时设置两个密码；生成随机密码；多次登陆失败锁定账户
- 多因子认证
- 授权相关的系统表，现在是InnoDB存储引擎。在 MySQL 8.0 之前，是 MyISAM 引擎



# 优化

- 引入了原生的，基于 InnoDB 的数据字典。数据字典表位于 mysql 库中，对用户不可见，同 mysql 库的其它系统表一样，保存在数据目录下的 mysql.libd 文件中。不再置于 mysql 目录下。
- 重构了 INFORMATION\_SCHEMA，其中，部分表已重构为基于数据字典的视图，在此之前，其为临时表
- PERFORMANCE\_SCHEMA 查询性能提升，其已内置多个索引
- Redo Log 的优化，包括允许多个用户线程并发写入 log buffer，可动态修改 innodb\_log\_buffer\_size 的大小
- 默认的内存临时表由 MEMORY 引擎更改为 TempTable 引擎，相比于前者，后者支持以变长方式存储 VARCHAR，VARBINARY 等变长字段。从 MySQL 8.0.13 开始，TempTable 引擎支持 BLOB 字段



## 其它

- 备份锁
- 可设置管理 IP 和端口，再也不用担心连接数满了，又登陆不上实例调整 max\_connections 的大小
- 引入了共享临时表空间 ibtmp1，用来存储临时表
- 升级无需执行 mysql\_upgrade



# MySQL 学习资料推荐

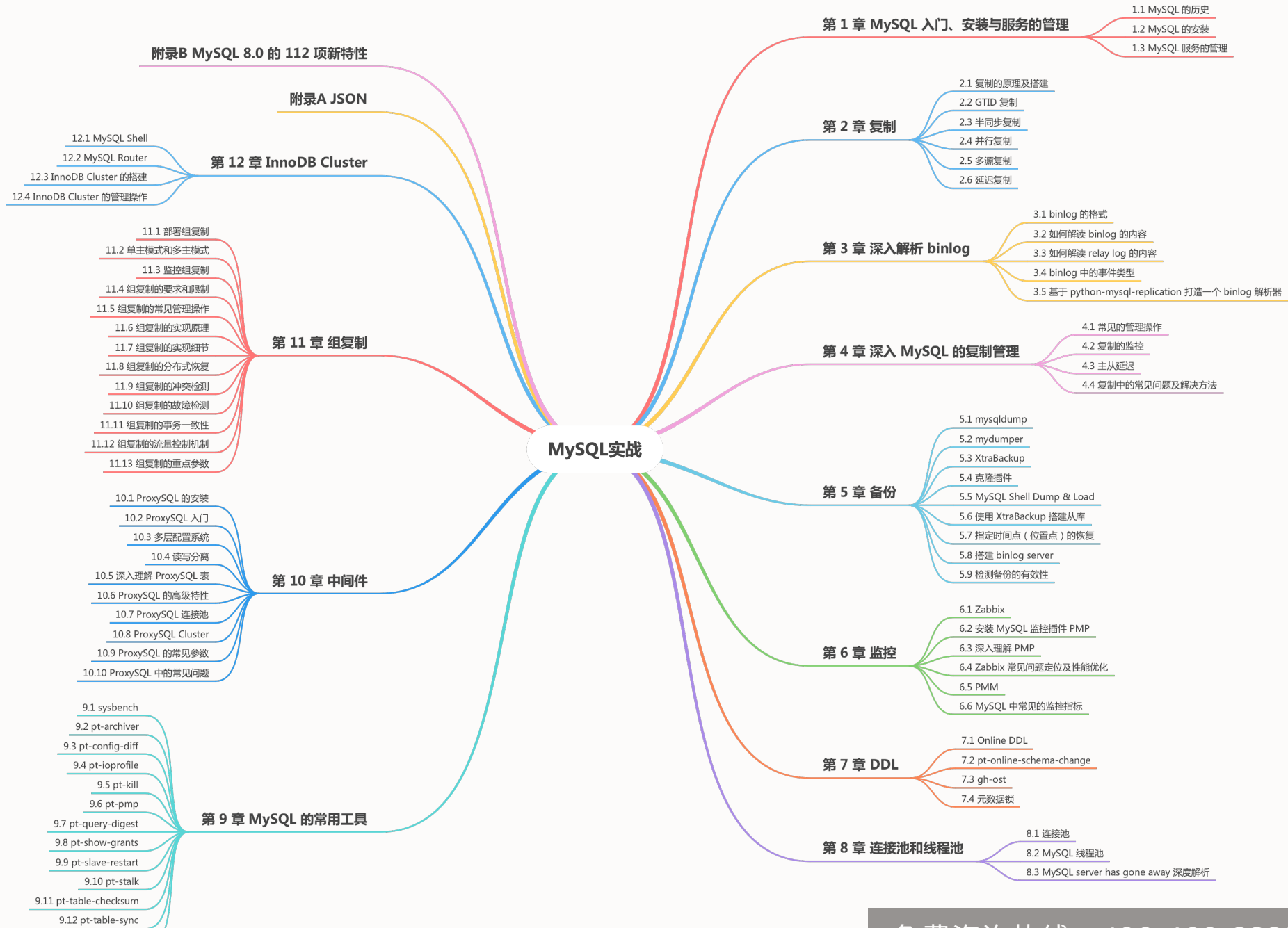
---

## MySQL 学习资料推荐

- 官方文档 - <https://dev.mysql.com/doc/refman/8.0/en/>
- Release Notes - <https://dev.mysql.com/doc/relnotes/mysql/8.0/en/>
- 官方博客 - <https://dev.mysql.com/blog-archive/>
- Worklog - <https://dev.mysql.com/worklog/>







# Thank you





# 基于 Oracle 数据库 免费企业数据健康检查

- 及时了解数据库健康状况，发现并解决潜在问题
- 维护数据库系统良好状态，保护数据资产的安全
- 提升数据库性能、稳定性和安全性，降低业务风险

免费咨询热线：

**400-699-8888**

\* 活动最终解释权归甲骨文公司所有

# Oracle Sharding 可扩展的多模型分布式 数据库



范宏伟

- 资深解决方案工程师
- 电信行业架构师
- 15年以上数据相关工作经验

## 内容简介

Oracle RAC和Data Guard可以满足数据库应用的绝大部分需求，但是仍然有一些用户的特别应用，需要更高的可扩展性和故障隔离性，因此Oracle从12.2开始支持Sharding部署模式。主要介绍sharding部署模式的主要组件、生命周期管理和技术演进。



Zoom直播

直播时间：7月14日 11:00 - 12:00

扫描二维码进入直播

Zoom ID: 976 6962 5763

密码: 98039717



微信扫一扫预约



数据库和云讲座群

20-21



甲骨文云技术公众号



技术专家1V1深入交流



ORACLE