系统架构设计师

数据库

姜美荣

目录

直播内容：案例之数据库

- 规范化与反规范化☆☆☆☆
- 索引与视图☆☆☆
- 分布式数据库☆☆☆☆
- NoSQL☆☆☆☆
- 内存数据库☆☆☆☆
- 数据库的优化☆☆☆☆

数据库考试内容

年份	主题	分值
2025上	Redis主从同步填空、增量主从同步填空、Redis数据库持久化的方式对比	25
2024年下	数据库设计–Cache–aside架构、数据库读写、一致性问题	25分
2024年上	数据库设计–分布式锁、Redis锁、死锁场景、ZSET命令	25分
2023年	数据库设计–数据持久层，Redis，读写分离	25分
	大数据架构	25分
2022年	分布式数据库缓存设计–一致性、布隆过滤器	25分
2021年	数据库设计–规范化与反规范化技术、Redis	25分
2020年	分布式数据库缓存设计–Redis	25分
	数据库设计–规范化数据库设计	25分
2019	分布式数据库缓存设计–一致性问题和运维	25分
	Web系统架构设计–SQL注入攻击	7分
2018	分布式数据库缓存设计–MemCache和Redis	25分

数据库考试内容

本章为核心章节，须反复练习。

单选：4分左右

案例：有所涉及

基本概念：三级模式-两级映像、关系代数运算

数据库模型：E-R模型、关系模型、关系代数（结合SQL语言）

规范化：函数依赖、键与约束、范式、模式分解

事务并发：并发三种问题、三级封锁协议

数据库的设计：规划、需求分析、概念设计、逻辑设计、物理设计。各阶段的可交付成果。数据库的性能优化：硬件、设计、应用软件、系统软件。

数据库新技术：数据库安全与备份、反规范化、分布式数据库、缓存数据库、数据库集群、两阶段提交、三阶段提交

Redis:主从复制、同步机制、三种模式、持久化方式

数据库设计

(1)需求分析：即分析数据存储的要求，产出物有数据流图、数据字典、需求说明书。获得用户对系统的三个要求：信息要求、处理要求、系统要求。

(2)概念结构设计：就是设计E-R图，也即实体-联系图。工作步骤包括：**选择局部应用、逐一设计分E-R图、E-R图合并。**

分E-R图进行合并时，它们之间存在的冲突主要有以下3类。

- 属性冲突。同一属性可能会存在于不同的分E-R图中。
- 命名冲突。相同意义的属性，在不同的分E-R图上有着不同的命名，或是名称相同的属性在不同的分E-R图中代表着不同的意义。
- 结构冲突。同一实体在不同的分E-R图中有不同的属性，同一对象在某一分E-R图中被抽象为实体而在另一分E-R图中又被抽象为属性。

(3)逻辑结构设计：将E-R图，转换成关系模式。工作步骤包括：**确定数据模型、将E-R图转换成为指定的数据模型、确定完整性约束和确定用户视图、规范化设计、优化。**

(4)物理设计：步骤包括确定**数据分布、存储结构和访问方式**。

(5)数据库实施阶段。根据逻辑设计和物理设计阶段的结果建立数据库，**编制与调试应用程序，组织数据入库，并进行试运行。**

(6)数据库运行和维护阶段。数据库应用系统经过试运行即可投入运行，但该阶段需要不断地对系统进行评价、调整与修改。数据库维护工作的主要内容包括**对数据库性能的监测和改善、故障恢复、数据库的重组和重构**。在数据库运行阶段，对数据库的维护主要由**DBA** 完成。

规范化理论

不规范化带来的四大问题：

设有一个关系模式R（SNAME，CNAME，TNAME TADDRESS），其属性分别表示学生姓名、选修的课程名、任课教师姓名和任课教师地址。仔细分析一下，就会发现这个模式存在下列存储异常的问题：

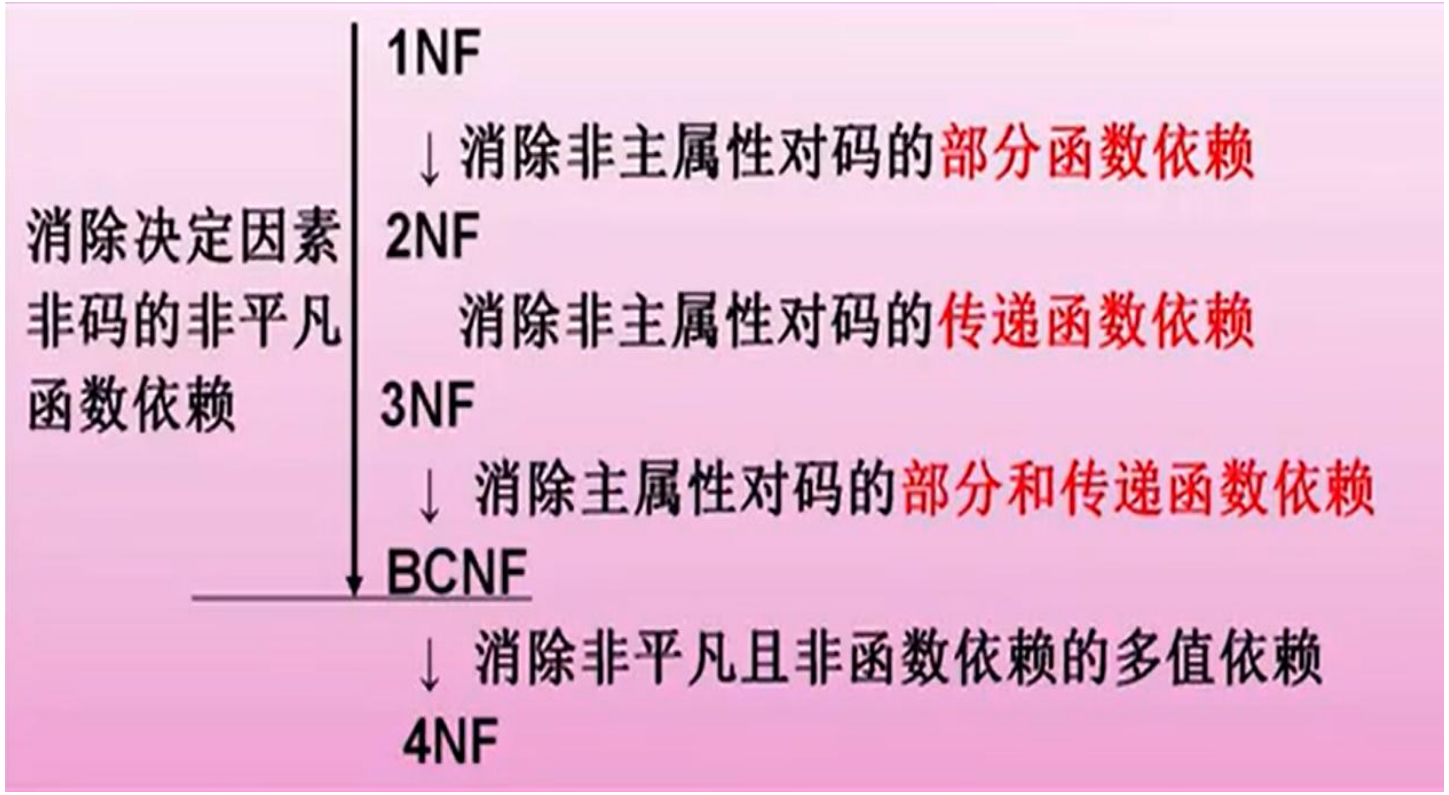
（1）**数据冗余：**数据被重复存储，如某门课程有100个学生选修，那么在R的关系中就要出现100个元组，这门课程的任课教师姓名和地址也随之重复出现100次。

（2）**修改异常：**修改导致数据不一致，如由于上述冗余问题，当需要修改这个教师的地址时，就要修改100个元组中的地址值，否则就会出现地址值不一致的现象。

（3）**插入异常：**插入时异常，如不知道听课学生名单，这个教师的任课情况和家庭地址就无法进入数据库；否则就要在学生姓名处插入空值。

（4）**删除异常：**删除了不该删除的数据，如当只有一条记录时，要删除这个学生选课信息，会将课程名、教师名和教师地址都给删除了。

规范化理论


1NF
↓ 消除非主属性对码的部分函数依赖
消除决定因素非码的非平凡函数依赖
2NF
消除非主属性对码的传递函数依赖
3NF
↓ 消除主属性对码的部分和传递函数依赖
BCNF
↓ 消除非平凡且非函数依赖的多值依赖
4NF

反规范化

反规范化是加速读操作性能(数据检素)的方法，通过有选择地在数据结构标准化后添加特定的冗余数据来增加查询效率，牺牲部分规范化来提高性能。

益处：连接操作少，**检索快、统计快**，需要查的表减少，检索容易。

挑战：空间占用大，操作开销大，数据不一致，更新和插入代码更难。

常见的反规范化操作有：**冗余列、派生列、表重组和表分割**，其中表分割又分为水平分割和垂直分割。

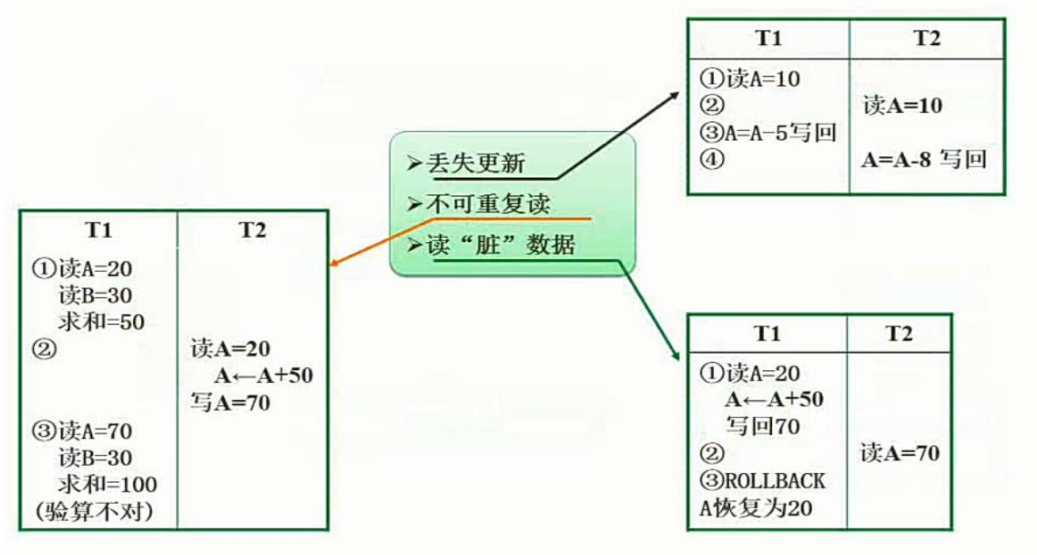
技术手段	说明
增加派生性冗余列	已有单价和数量列，增加“总价”列
增加冗余列	已有学号列，增加“姓名”列
重新组表	把拆分的表重新组表
分割表	把用户表做水平分割，长沙的用户存在长沙，上海的用户存在上海

反规范化的缺点	解决方案
数据冗余，需要更大存储空间	无解
插入、更新、删除操作开销更大	无解
数据不一致 可能产生添加、修改、删除异常	1、触发器数据同步 2、应用程序数据同步 3、批处理
更新和插入代码更难写	无解

并发控制：事务管理

事务通常以 BEGIN TRANSACTION（事务开始）语句开始，以 COMMIT 或 ROLLBACK 语句结束。COMMIT 称为“事务提交语句”，表示事务执行成功的结束。ROLLBACK 称为“事务回退语句”，表示事务执行不成功的结束。

- **丢失更新**：事务1对数据A进行了修改并写回，事务2也对A进行了修改并写回，此时事务2写回的数据会覆盖事务1写回的数据，就丢失了事务1对A的更新。即对数据A的更新会被覆盖。
- **不可重复读**：事务2读A,而后事务1对数据A进行了修改并写回，此时若事务2再读A，发现数据不对。即一个事务重复读A两次，会发现数据A有误。
- **读脏数据**：事务1对数据A进行了修改后，事务2读数据A，而后事务1回滚，数据A恢复了原来的值，那么事务2对数据A做的事是无效的，读到了脏数据。



并发控制–事务管理–封锁技术

处理并发控制的主要方法是采用封锁技术。

它有两种类型：排他型封锁（X 封锁）和共享型封锁（S 封锁），分别介绍如下：

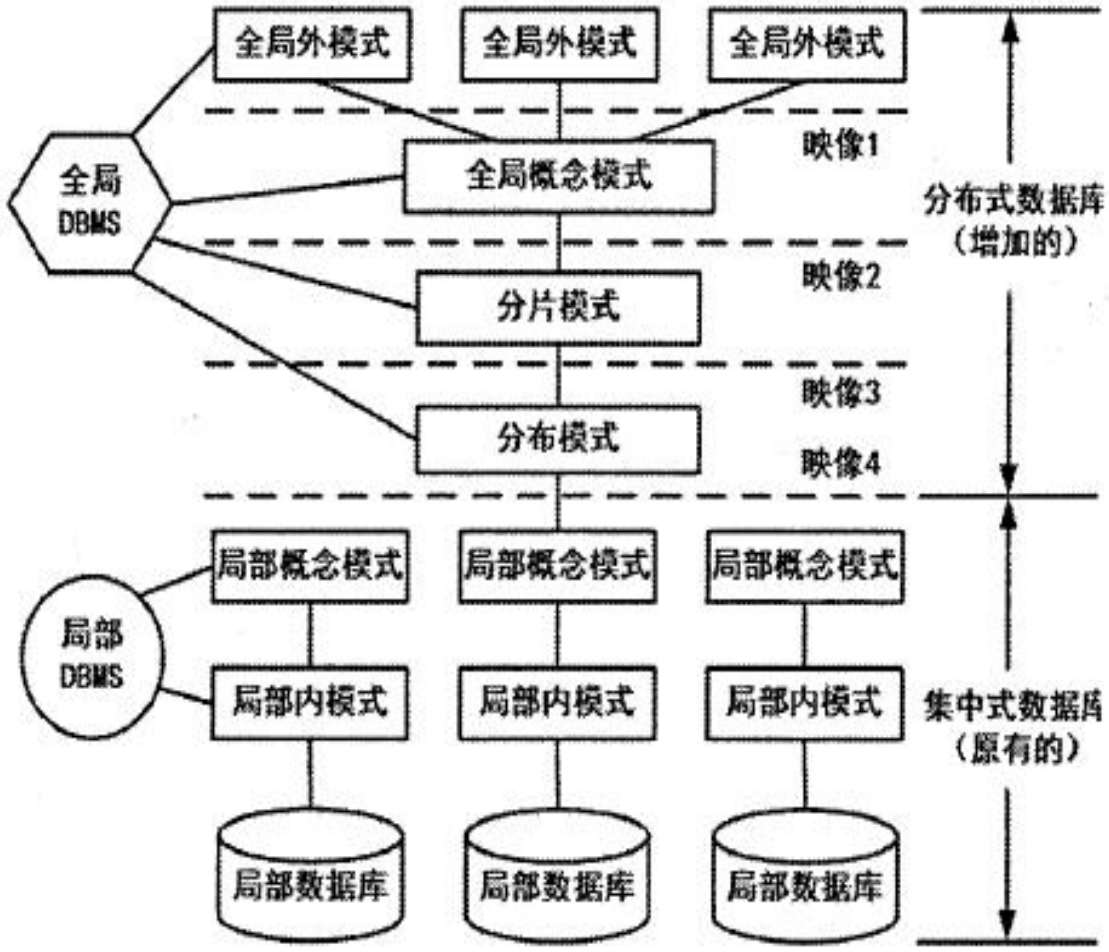
（1）排他型封锁（简称 X 封锁）。如果事务 T 对数据 A（可以是数据项、记录、数据集，乃至整个数据库）实现了 X 封锁，那么只允许事务 T 读取和修改数据 A，其他事务要等事务 T 解除 X 封锁以后，才能对数据 A 实现任何类型的封锁。可见 X 封锁只允许一个事务独锁某个数据，具有排他性。

（2）共享型封锁（简称 S 封锁）。X 封锁只允许一个事务独锁和使用数据，要求太严。需要适当放宽，例如可以允许并发读，但不允许修改，这就产生了 S 封锁概念。S 封锁的含义是：如果事务 T 对数据 A 实现了 S 封锁，那么允许事务 T 读取数据 A，但不能修改数据 A，在所有 S 封锁解除之前绝不允许任何事务对数据 A 实现 X 封锁。

独占X，共享S

分布式数据库管理系统

分布式数据库系统的模式结构有六个层次，可以概括和说明任何分布式数据库系统的概念和结构。



分布透明性

- **分片透明：即如何分片对用户是透明的。**
- **复制透明：用户不用关心数据库在网络中各个节点的复制情况。**
- **位置透明：是指用户不必知道所操作的数据放在何处。**
- **局部映像透明性(逻辑透明):用户不必关心局部DBMS支持哪种数据模型、使用哪种数据操纵语言，数据模型和操纵语言的转换是由系统完成的。**4种形式：
  - 水平分片：把全局关系的所有元组划分为不想交的子集。
  - 垂直分片：把全局关系的所有属性划分为若干子集，每个属性至少映射到一个垂直分片中，并且包含主键。
  - 导出分片：水平分片的条件不是本关系的属性条件，而是其他关系的属性条件。

分布式数据库管理系统

两阶段提交协议（★★）
2PC协议把数据提交分成两个阶段
- 表决阶段：协调者发布“准备提交”指令，所有参与者进行表决，具有一票否决权。目的是形成一个共同的决定。
- 执行阶段：根据协调者的指令，参与者提交或撤销事务，并给协调者发送确认信息。目的是实现这个协调者的决定。

规则：只要有一个参与者撤销事务，协调者就必须做出全局撤销的决定。
只有所有参与者都同意提交事务，协调者才能做出全局提交的决定。

三阶段提交协议
3PC协议把数据提交分成两个阶段
- 第一阶段：协调者向所有参与者发布“准备提交”指令，所有参与者进行表决，只有所有参与者都投票“建议提交”，才会进入第二阶段。
- 第二阶段：协调者向所有参与者发“全局预提交”，只有所有参与者都提交“准备就绪”才能进入第三阶段。
- 第三阶段：协调者向所有的参与者发布“全局提交”报文。

NOSQL VS 关系数据库

对比维度	关系数据库	NoSQL
应用领域	面向**通用**领域	**特定**应用领域
数据容量	**有限**数据	**海量**数据
数据类型	**结构化**数据二维表	**非结构化**数据
并发支持	支持**并发**、但**性能低**	**高并发**
事务支持	**高**事务性	**弱**事务性
扩展方式	向**上**扩展	向**外**扩展

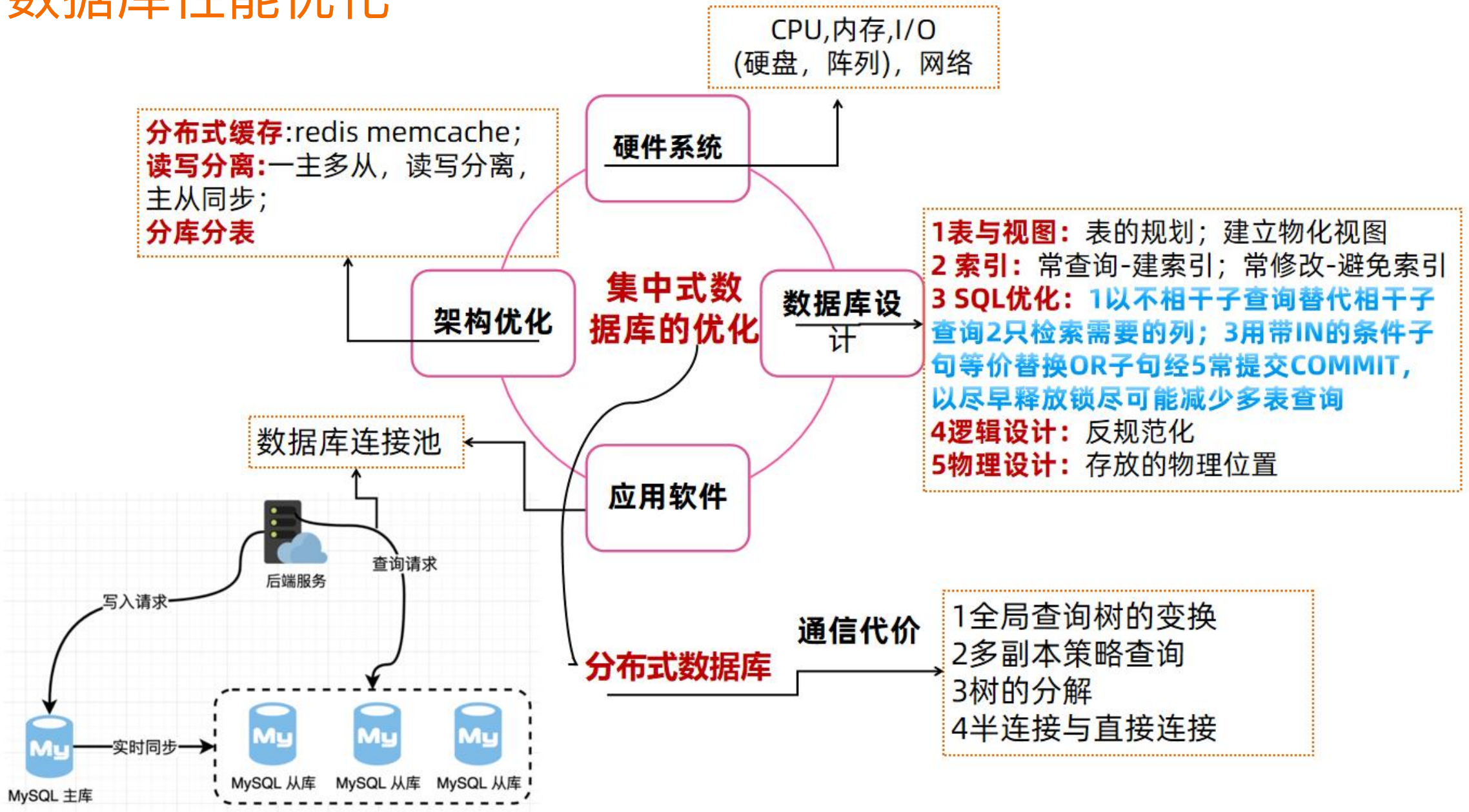
NoSQL 数据库

分　类	典型产品	应用场景	优　点	缺　点	数据模型
文档存储	**MongoDB**、CouchDB	Web应用，存储面向文档和半结构化数据	结构灵活，可以根据value构建索引	缺乏统一的查询语法，无事务处理能力	Key-Value对应的键值对，Value为结构化数据
键值存储	Memcached、**Redis**	内容缓存，如会话、配置文件、参数等	扩展性好，灵活性强，大量操作时性能高	数据无结构化，通常被当成字符串或者二进制数据，通过键查询值	Key 指向 Value的键值对，通常用hash table来实现
列存储	BigTable、**HBase**、Cassandra	分布式数据存储和管理	可扩展性强，查找速度快，复杂性低	功能局限，不支持事务的强一致性	以**列簇式存储**，将同一列数据存在一起
图存储	Neo4j、OrientDB	社交网络、推荐系统，专注于构建系统图谱	支持复杂的图形算法	复杂性高，只能支持一定的数据规模	图结构

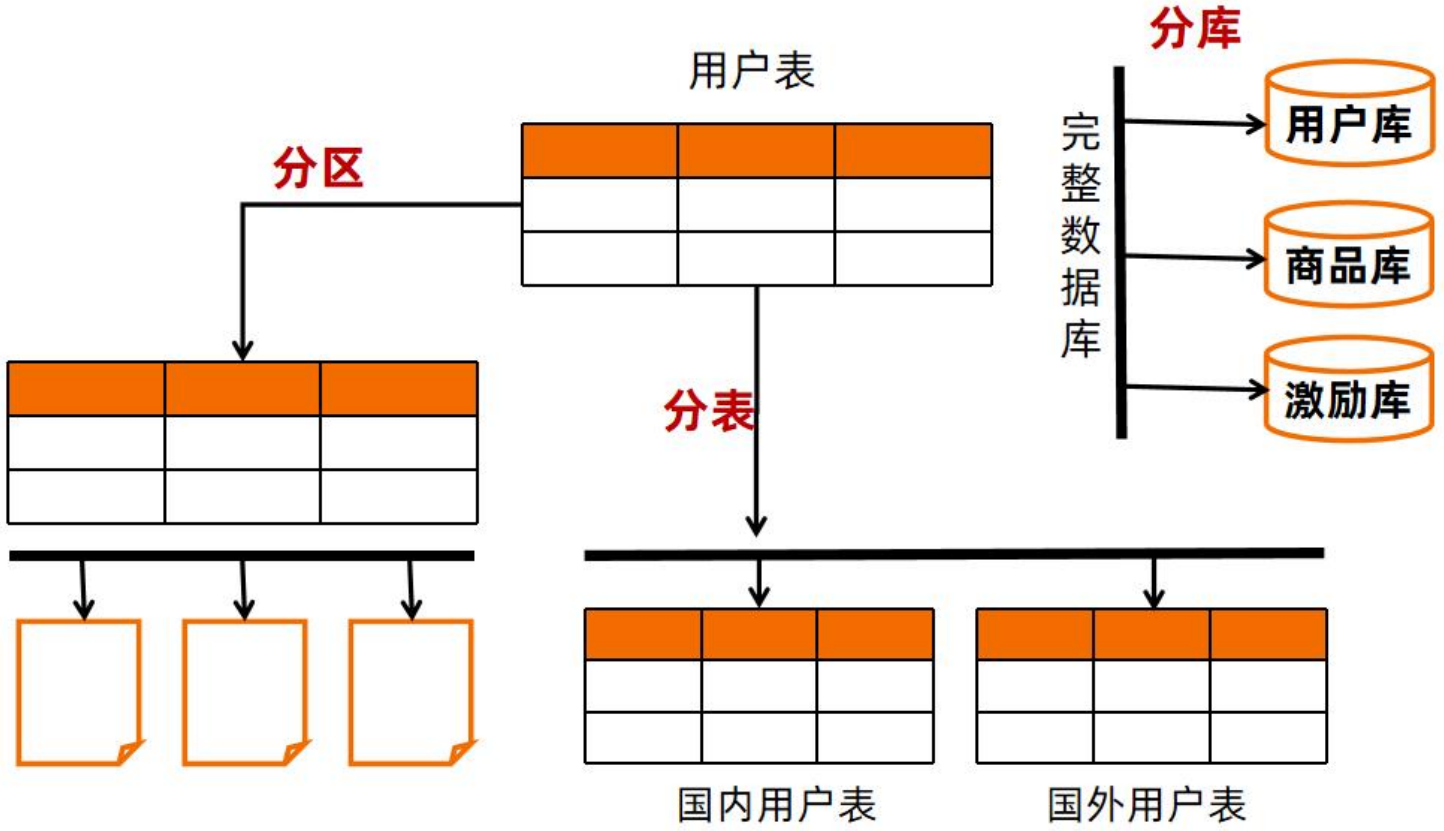
CAP理论

主要概念	解释
C（Consistency）一致性	一致性是指更新操作成功并返回客户端完成后，所有节点在同一时间的数据完全一致，与ACID的C完全不同。
A（Availability）可用性	可用性是指服务一直可用，而且是正常响应时间。
P（Partition tolerance）分区容忍性	分区容忍性是指分布式系统在遇到某节点或网络分区故障的时候，仍然能够对外提供满足一致性和可用性的服务。

数据库性能优化


CPU,内存,I/O
(硬盘，阵列)，网络
硬件系统
分布式缓存:redis memcache;
读写分离:一主多从，读写分离，
主从同步;
分库分表
架构优化
集中式数
据库的优化
数据库设
计
1表与视图：表的规划；建立物化视图
2 索引：常查询-建索引；常修改-避免索引
3 SQL优化：1以不相干子查询替代相干子
查询2只检索需要的列；3用带IN的条件子
句等价替换OR子句经5常提交COMMIT，
以尽早释放锁尽可能减少多表查询
4逻辑设计：反规范化
5物理设计：存放的物理位置
数据库连接池
应用软件
后端服务
查询请求
写入请求
分布式数据库
通信代价
1全局查询树的变换
2多副本策略查询
3树的分解
4半连接与直接连接
实时同步
MySQL 主库
MySQL 从库
MySQL 从库
MySQL 从库

分区分表分库



共性：

1、都针对数据表
2、都使用了分布式存储
3、都提升了查询效率
4、都降低了数据库的频繁I/O压力值

特性：

分区逻辑上还是一张表，
分表逻辑上已是多张表

分区分表分库–分区方法

Range（范围）　　按照时间拆分　　Hash之后按照分表个数取模

分区的优点

1、相对于单个文件系统或是硬盘，分区可以存储更多的数据。

2、数据管理比较方便，比如要清理或废弃某年的数据，就可以直接删除该日期的分区数据即可。

3、精准定位分区查询数据，不需要全表扫描查询，大大提高数据检索效率。

4、可跨多个分区磁盘查询，来提高查询的吞吐量。

5、在涉及聚合函数查询时，可以很容易进行数据的合并。

分区分表分库–分区方法

Range（范围）　　按照时间拆分　　Hash之后按照分表个数取模

分区的优点

1、相对于单个文件系统或是硬盘，分区可以存储更多的数据。

2、数据管理比较方便，比如要清理或废弃某年的数据，就可以直接删除该日期的分区数据即可。

3、精准定位分区查询数据，不需要全表扫描查询，大大提高数据检索效率。

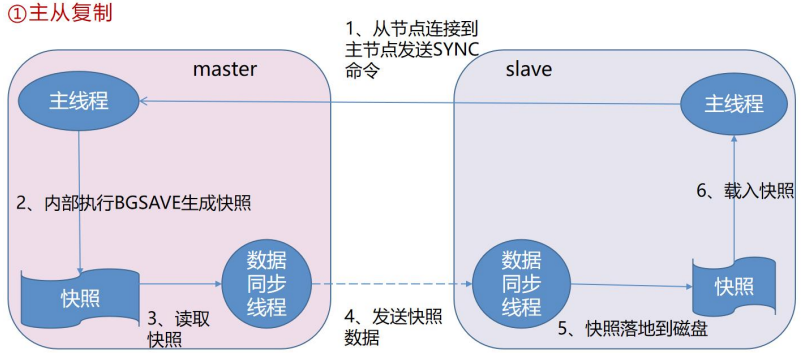
4、可跨多个分区磁盘查询，来提高查询的吞吐量。

5、在涉及聚合函数查询时，可以很容易进行数据的合并。

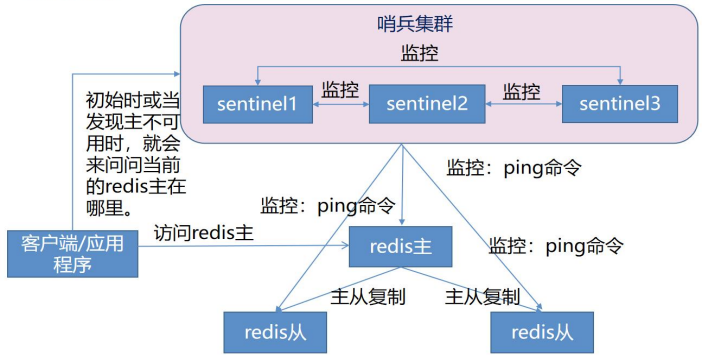
Redis分布式存储方案

Redis分布式存储方案	模式
主从模式	一主多从，故障时手动切换。
哨兵模式	有哨兵的一主多从，主节点故障**自动选择**新的主节点。
集群模式	**分节点对等集群**，分**slots**，不同slots的信息存储到不同节点

①主从复制



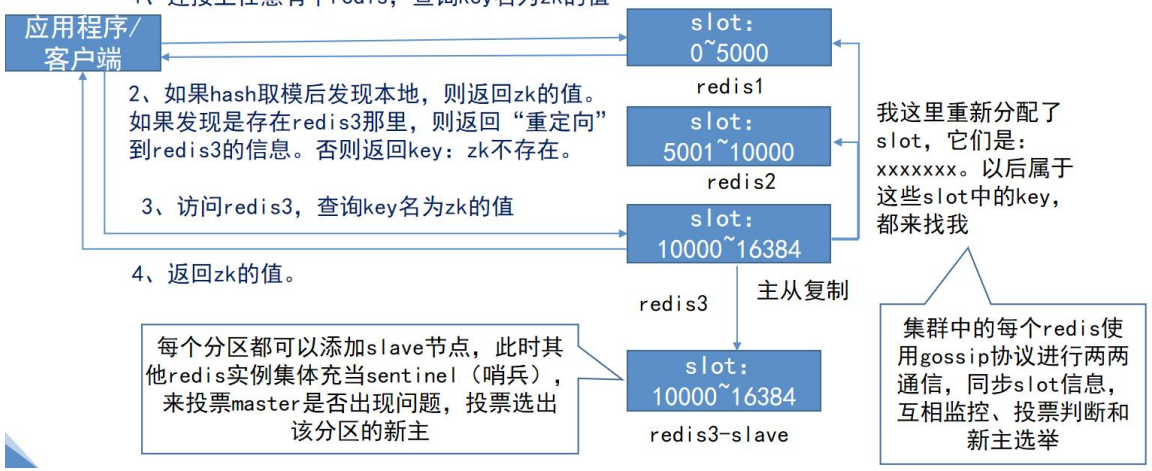
②哨兵集群



③cluster集群

所有的key，经过CRC16运算后，再对16384取模（取余），散列到16384个slot中。
每个redis承担一定数量的slot（哈希分区、范围分区结合应用）

每个redis都维护着全局slot和redis的对应关系，都知道查询某个key应该找哪个redis。



我这里重新分配了slot，它们是：xxxxxxx。以后属于这些slot中的key，都来找我

每个分区都可以添加slave节点，此时其他redis实例集体充当sentinel（哨兵），来投票master是否出现问题，投票选出该分区的新主

集群中的每个redis使用gossip协议进行两两通信，同步slot信息，互相监控、投票判断和新主选举

Redis持久化存储的机制

Redis提供了两种持久化存储的机制，分别是**RDB (Redis DataBase)持久化方式和AOF (Append Only File)持久化方式。**一旦服务器宕机，内存中的数据将全部丢失。我们很容易想到的一个解决方案是，从后端数据库恢复这些数据，但这种方式存在两个问题：一是需要频繁访问数据库，会给数据库带来巨大的压力；二是这些数据是从慢速数据库中读取出来的，性能肯定比不上从 Redis 中读取，导致使用这些数据的应用程序响应变慢。所以，对 Redis 来说，实现数据的持久化，避免从后端数据库中进行恢复，是至关重要的。（★★）

两种持久化机制	RDB内存快照 （RedisDataBase）	AOF日志 （Append Only Flie）
说明	把当前内存中的数据集快照写入磁盘（数据库中所有键值对数据）。恢复时是将快照文件直接读到内存里。	通过持续不断地保存Redis服务器所执行的更新命令来记录数据库状态，类似mysql的binlog。恢复数据时需要从头开始回放更新命令。
磁盘刷新频率	低	高
文件大小	小	大
数据恢复效率	高	低
数据安全	低	高

典型真题

阅读下列说明，回答问题1至问题3,将解答填入答题纸的对应栏内。
某企业委托软件公司开发一套包裹信息管理系统，以便于对该企业通过快递收发的包裹信息进行统一管理。在系统设计阶段，需要对不同快递公司的包裹单信息进行建模，其中，邮政包裹单如图2-1所示。

【问题1】(14分)
请说明关系型数据库开发中，逻辑数据模型设计过程包含哪些任务?该包裹单的逻辑数据模型中应该包含哪些实体?并给出每个实体的主键属性。
【问题2】(6分)
请说明什么是超类实体?结合图中包裹单信息，试设计一种超类实体，给出完整的属性列表 。
【问题3】(5分)
请说明什么是派生属性，并结合图2-1的包裹单信息说明哪个属性是派生属性。

图 2-1　包裹单示意图

典型真题

【问题1】
逻辑数据模型设计过程包含的任务：
(1)构建系统上下文数据模型，包含实体及实体之间的联系。
(2)绘制基于主键的数据模型，为每个实体添加主键属性。
(3)构建全属性数据模型，为每个实体添加非主键属性。
(4)利用规范化技术建立系统规范化数据模型。
包裹单的逻辑数据模型中包含的实体：
(1)收件人(主键：电话)。
(2)寄件人(主键：电话)。
(3)包裹单(主键：编号)。
【问题2】
超类实体是将多个实体中相同的属性组合起来构造出的新实体。 用户(姓名、电话、单位名称、详细地址)
【问题3】
派生属性是指某个实体的非主键属性由该实体其他非主键属性决定。
包裹单中的总计是由资费、挂号费、保价费、回执费计算得出，所以是派生属性。

典型真题

阅读以下关于数据管理的叙述，在答题纸上回答问题1至问题3。

【说明】

某软件企业开发了一套新闻社交类软件，提供常见的新闻发布、用户关注、用户推荐、新闻点评、新闻推荐、热点新闻等功能，项目采用MySQL数据库来存储业务数据。系统上线后，随着用户数量的增加，数据库服务器的压力不断加大。为此，该企业设立了专门的工作组来解决此问题。

张工提出对MySQL数据库进行扩展，采用读写分离，主从复制的策略，好处是程序改动比较小，可以较快完成，后续也可以扩展到MySQL集群，其方案如图4-1所示。李工认为该系统的诸多功能，并不需要采用关系数据库，甚至关系数据库限制了功能的实现，应该采用 NoSQL数据库来替代MySQL，重新构造系统的数据层。而刘工认为张工的方案过于保守，对该系统的某些功能，如关注列表、推荐列表、热搜榜单等实现困难，且性能提升不大；而李工的方案又太激进，工作量太大，短期无法完成，应尽量综合二者的优点，采用Key-Value数据库+MySQL数据库的混合方案。

经过组内多次讨论，该企业最终决定采用刘工提出的方案。

典型真题

【问题1】（8分）
张工方案中采用了读写分离，主从复制策略。其中，读写分离设置物理上不同的主/从服务器，让主服务器负责数据的（a）操作，从服务器负责数据的（b）操作，从而有效减少数据并发操作的（c），但却带来了（d）。因此，需要采用主从复制策略保持数据的（e）。
My SQL 数据库中，主从复制是通过 binary log来实现主从服务器的数据同步， My SQL 数据库支持的三种复制类型分别是（f）、（g）、（h）。
请将答案填入（a）~（h）处的空白，完成上述描述。

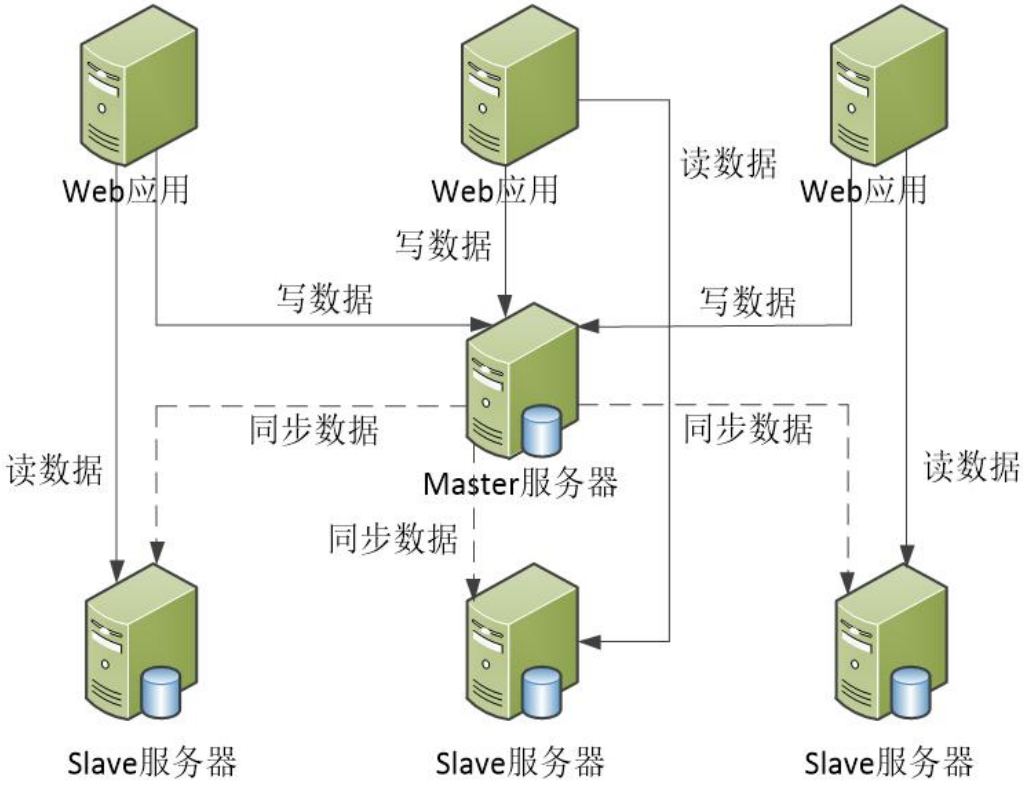


图 4-1 张工方案示意图

典型真题

【参考答案】
（a）写（b）读（c）延迟（d）数据冗余（e）一致性（或同步）（f）基于SQL语句的复制（g）基于行的复制（h）混合模式复制
注：（f）（g）（h）不分次序。

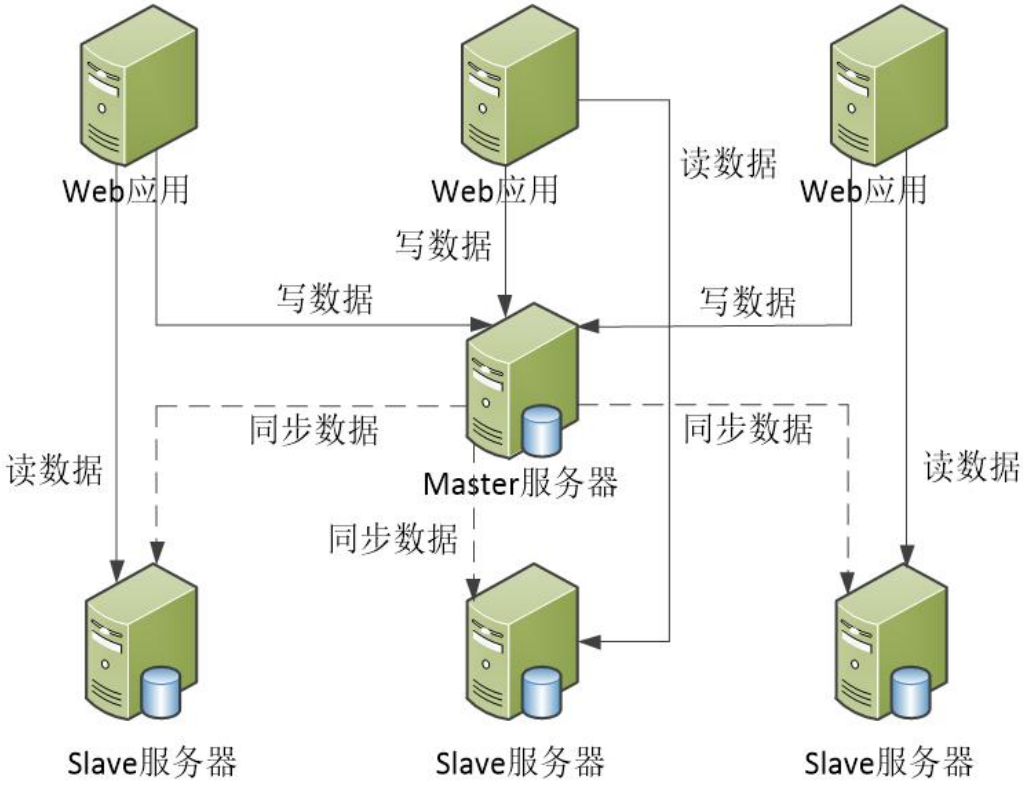


图 4-1 张工方案示意图

典型真题

【问题2】（3分）

李工方案中给出了关系数据库与No SQL数据库的比较，如表4-1所示，以此来说明该新闻社交类软件更适合采用No SQL数据库。请完成表 4-1 中的（a）~（d）处空白。

表 4-1 关系数据库与 No SQL 数据库特征比较

特征	关系数据库	No SQL 数据库
数据一致性	实时一致性	（a）
数据类型	结构化数据	（b）
事务	高事务性	（c）
水平扩展	弱	强
数据容量	有限数据	（d）

典型真题

【参考答案】
（a）最终一致性
（b）非结构化数据
（c）软状态/柔性事务
（d）海量数据

典型真题

【问题3】（3分）

刘工提出的方案采用了Key-Value数据库+MySQL数据库的混合方案，是根据数据的读写特点将数据分别部署到不同的数据库中。但是由于部分数据可能同时存在于两个数据库中，因此存在数据同步问题。请用 200 字以内的文字简要说明解决该数据同步问题的三种方法。

典型真题

【问题3】（3分）

刘工提出的方案采用了Key-Value数据库+MySQL数据库的混合方案，是根据数据的读写特点将数据分别部署到不同的数据库中。但是由于部分数据可能同时存在于两个数据库中，因此存在数据同步问题。请用 200 字以内的文字简要说明解决该数据同步问题的三种方法。

【参考答案】

（1）采用自定义函数，进行MySQL数据库编程，利用触发器实现数据同步。

（2）采用消息中间件实现数据同步。

（3）采用同步工具，比如canal实现数据同步。

作业（202505真题综合技术）

7．关系代数运算不包括（）。

A.选择　　B.投影　　C.删除　　D.连接

【解析】：

关系代数运算符有4 类：集合运算符、专门的关系运算符、算术比较符和逻辑运算符。

【参考答案】：C

32.数据库三级模式中，表示用户局部数据的是（）。

A.内模式　　B.外模式　　C.概念模式　　D.逻辑模式

【解析】：

外模式也称用户模式或子模式，是用户与数据库系统的接口，是用户需要使用的部分数据的描述。它由若干个外部记录类型组成。用户使用数据操纵语言对数据库进行操作，实际上是对外模式的外部记录进行操作。

【参考答案】：B

作业（202505真题综合技术）

33.数据库三级模式中，描述记录间联系、数据完整性、安全性的是（）。

A.存储模式　　B.概念模式　　C.内模式　　D.外模式

【解析】：

概念模式也称模式，是数据库中全部数据的逻辑结构和特征的描述，它由若干个概念记录类型组成，只涉及“型”的描述，不涉及具体的值。概念模式的一个具体值称为模式的一个实例，同一个模式可以有很多实例。概念模式反映的是数据库的结构及其联系，所以是相对稳定的；而实例反映的是数据库某一时刻的状态，是相对变动的。

需要说明的是，概念模式不仅要描述概念记录类型，还要描述记录间的联系、操作、数据的完整性和安全性等要求。但是，概念模式不涉及存储结构、访问技术等细节。只有这样，概念模式才算做到了“物理数据独立性”。

【参考答案】：B

作业（202505真题综合技术）

34.已知关系R(a,b,c,d)和R上的函数依赖F=(a->cd，c->b)，则R的候选码是（）。

A.a　　B.b　　C.c　　D.d

【解析】：

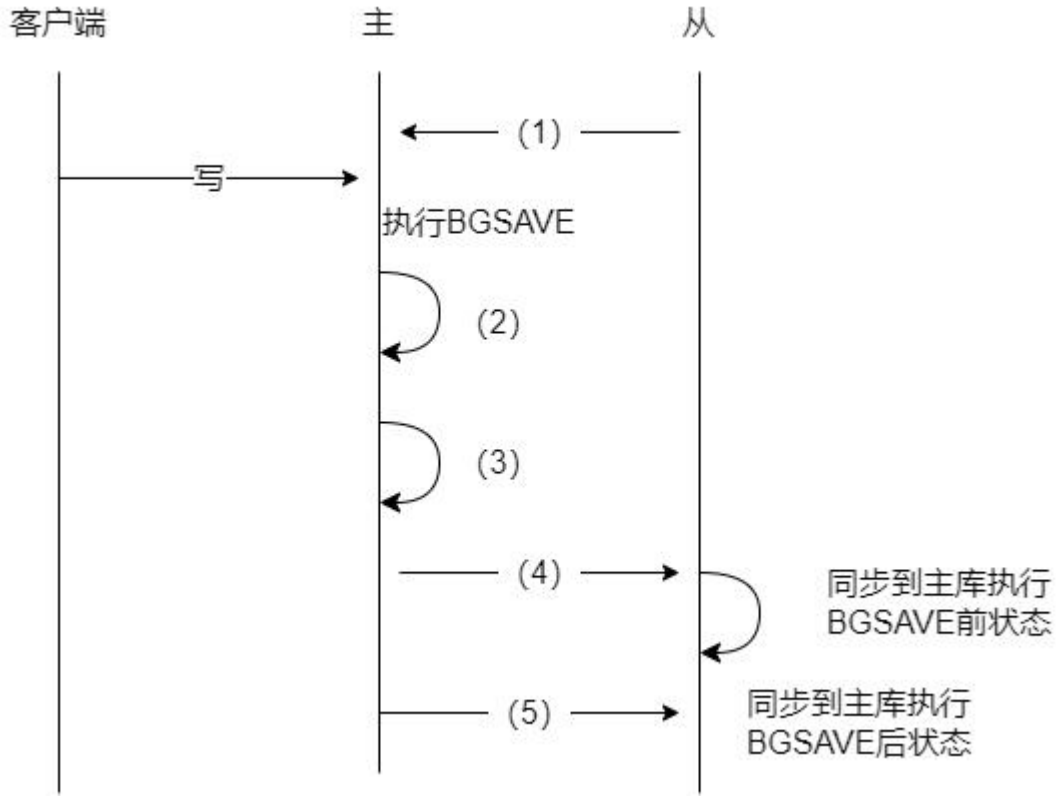
已知a→cd，推导出a可确定c和d；

进一步通过c→b，可得a→b，因此闭包为a = U（全体属性），候选码是a。

【参考答案】：A

作业（202505真题）

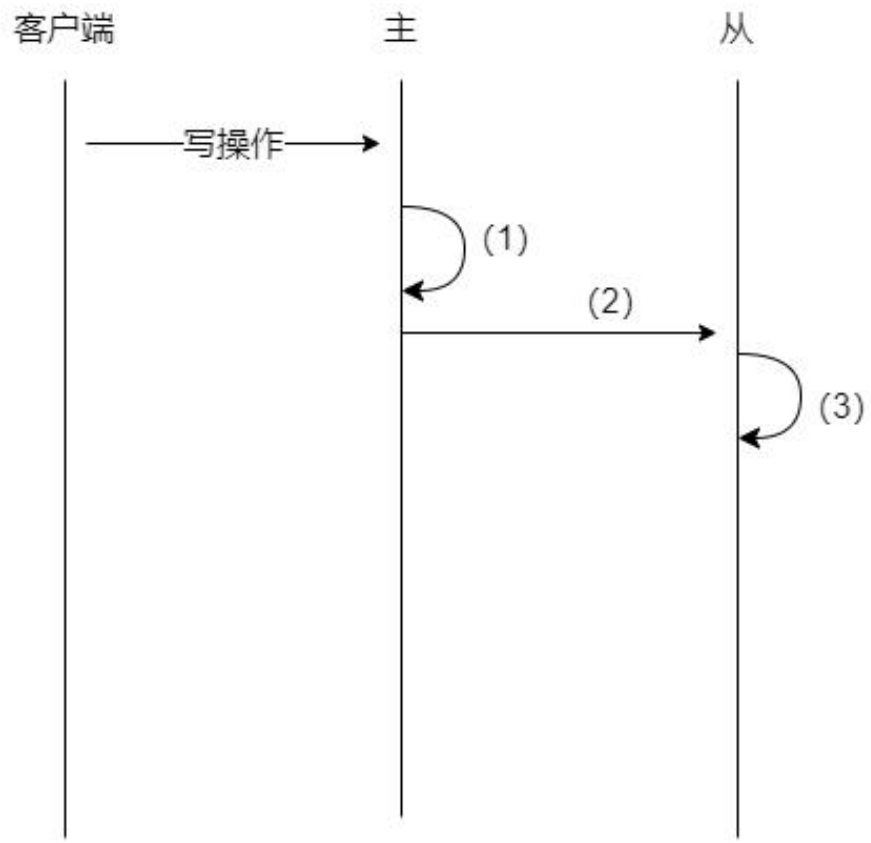
1、（10分）Redis 主从同步，按要求填空，首次全量从库同步主库过程。



作业（202505真题）

2、（6分）增量主从库同步。

增量主从库同步



作业（202505真题）

3、Redis数据库持久化的方式，列举两种并作比较，说明两种持久化方式的优缺点。（9分）

作业答案

1【参考答案】：
（1）请求数据同步
（2）判断是否全量同步
（3）将操作记录在repl_baklog内
（4）发送RDB文件
（5）发送repl_baklog中的命令
【解析】：
数据同步阶段（含 BGSAVE 操作 ）
（1）同步请求：从节点发送psync或psync2命令给主节点，请求进行数据同步。初次同步时，从节点发送的psync命令中，replid和offset的默认值分别是?和-1，表示需要进行全量复制 。
（2）主节点执行 BGSAVE：主节点接收到从节点的同步请求后，判断需要进行全量复制，会 fork 一个子进程来执行BGSAVE命令 。BGSAVE用于在后台生成 RDB 快照文件，这个文件保存了主节点当前时刻的数据状态。在执行BGSAVE期间，主节点会开辟一个命令缓冲区，用来记录在此期间接收到的写操作指令 。
（3）生成并发送 RDB 文件：BGSAVE子进程生成 RDB 文件后，主节点通过 socket 连接将 RDB 文件发送给从节点 。
（4）从节点接收并处理 RDB 文件：从节点接收到 RDB 文件后，会清空自己以前的数据，然后执行 RDB 文件的恢复过程，将自身数据库状态更新至主节点执行BGSAVE前的状态 。
（5）同步写命令缓冲区：从节点完成 RDB 文件恢复后，会向主节点发送信息告知恢复已完成。主节点将在执行BGSAVE期间记录在命令缓冲区中的写命令，以 AOF（Append Only File）的格式发送给从节点 。从节点接收这些命令并执行，使其数据库状态同步到主节点执行BGSAVE后的状态 。如果从节点开启了 AOF 持久化功能，它会进行bgrewriteaof操作，生成最新的 AOF 文件，以保证数据的持久性和安全性 。

作业答案

2【参考答案】：

（1）检查数据更改

（2）发送offset之后的数据

（3）执行命令

作业答案

3【参考答案】：
1. RDB（Redis Database）
优点：
高性能：仅在触发快照时 fork 子进程进行持久化，不影响主进程性能。
恢复快：完整二进制文件，重启加载速度显著快于 AOF。
节省空间：紧凑二进制格式，适合备份、灾难恢复。
缺点：
数据易丢失：默认触发频率为 5 分钟一次（可配置），若在此期间宕机，可能丢失最新数据。
fork 开销：大数据量下 fork 子进程可能导致短暂阻塞（毫秒级）。
2. AOF（Append Only File）
优点：
数据安全性高：支持秒级持久化（everysec 策略），最多丢失 1 秒数据。
实时记录：记录写操作命令，可通过回放日志恢复数据。
兼容性强：即使日志文件损坏，也可通过 Redis-check-aof 工具修复。
缺点：
文件体积大：记录所有写命令，文件增长速度快于 RDB。
性能开销：频繁写磁盘（尤其 always 策略）可能降低 QPS。
恢复较慢：需逐条执行命令回放，大数据量下耗时较长。
对比选择建议：
优先选 RDB：若业务对数据丢失容忍度高（如缓存场景），追求高性能恢复。
优先选 AOF：若需保证数据完整性（如支付、账户系统），可接受稍低性能。
最佳实践：二者同时启用，兼顾数据安全性与恢复效率。

数据持久化

Redis 提供两种主要的持久化机制（ps：**持久化是为了快速的恢复数据而不是为了存储数据**）：

持久化机制	说明	优点	缺点	适用场景
RDB（Redis Database）	在指**定的时间间隔内，将内存中的数据集快照写入磁盘**，恢复时将快照文件直接读到内存。	1. 适合大规模数据恢复 2. **对性能影响小**，fork子进程进行持久化，主进程继续处理请求 3. **文件紧凑**，适合备份和灾难恢复	1. **可能会丢失最后一次快照后的数据** 2. fork子进程时，内存不足可能影响性能 3. 保存RDB文件时，**如果数据集大，时间会较长**	1. **可以容忍一定数据丢失** 2. **大规模数据恢复场景** 3. 对数据恢复速度要求较高
AOF（Append Only File）	**以日志形式记录每个写操作，追加到AOF文件**，重启时重新执行这些命令恢复数据。	1. **数据安全性高**，可配置不同同步策略，最多丢失1秒数据 2. AOF文件是**有序的写操作日志**，易读，可修改修复	1. **AOF文件通常比RDB文件大** 2. **性能比RDB稍差**，因为要不断记录写操作 3. 恢复速度比RDB慢	1. **对数据安全性要求高**，不能容忍数据丢失 2. **对恢复速度要求不苛刻**
RDB + AOF（版本4.0新增的机制）	**同时使用RDB和AOF持久化。**	兼具RDB和AOF的优点，既有RDB的快速恢复能力，又有AOF的数据安全性。	1. 占用更多磁盘空间 2. 配置和管理相对复杂	**对数据安全性和恢复速度都有较高要求的场景**

THANKS