数据库系统概论总结

李鹏

华东师范大学数据学院

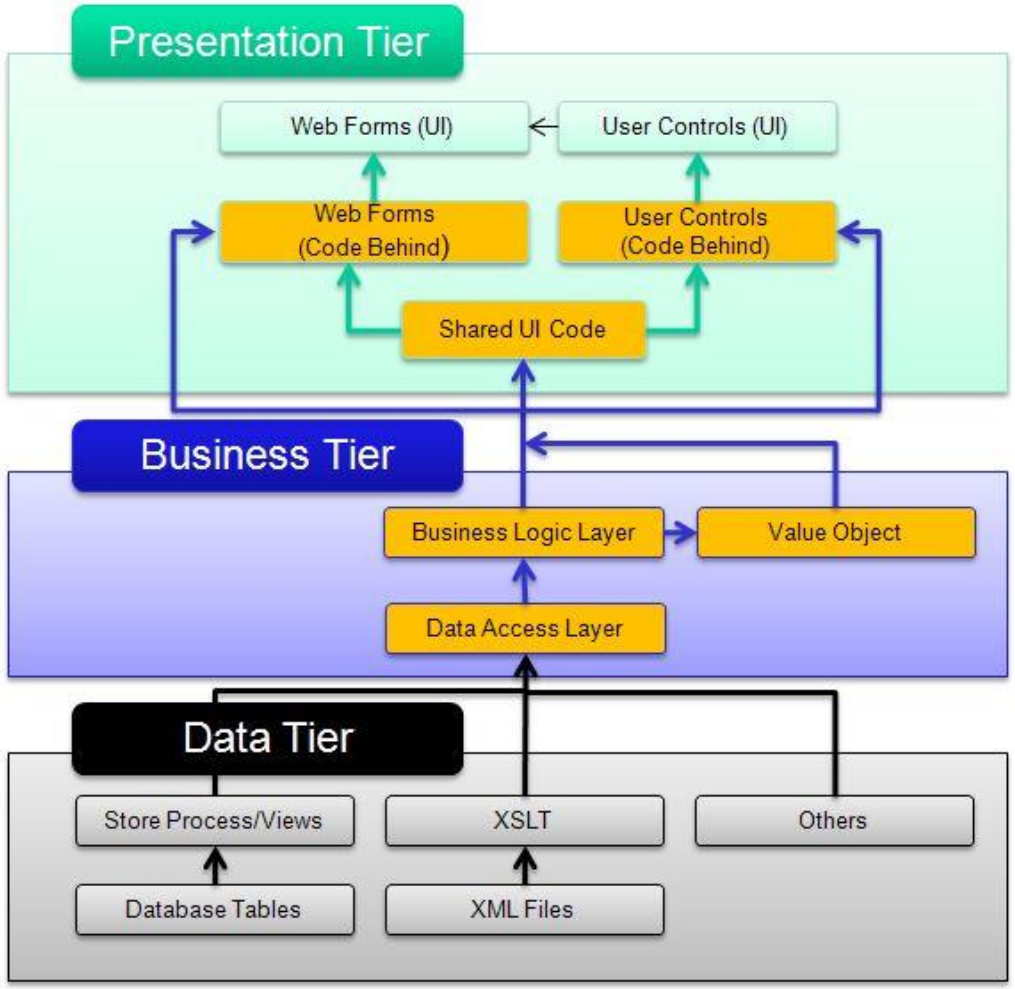
04/11/2020

什么是数据库系统

用户需求

应用软件

系统软件



数据库系统的发展

- 网状/层次数据库
- SQL数据库
- NoSQL数据库
- NewSQL数据库

提纲：SQL/NoSQL/NewSQL

- 数据库的设计思想
  - 数据库为什么长这样
- 数据库的使用
  - 增删改查
- 重要的内部实现细节
  - 存储/查询/事务/可用性…
- 设计练习

提纲：SQL/NoSQL/NewSQL

- 数据库的设计思想
  - 数据库为什么长这样
- 数据库的使用
  - 增删改查
- 重要的内部实现细节
  - 存储/查询/事务/可用性…
- 设计练习

数据模型

- 数据模型→数据的访问方式→DBMS的访问接口→系统的功能性、性能、易用性
- 数据模型：网状？层状？关系？
- 数据模型的表达能力（范围/简易程度）
  - 能够满足查询的需求的能力
  - 声明式程序设计语言SQL

关系模型

- 域
- 笛卡尔积
- 关系（键）
- 关系代数
  - 集合运算符：交/差/并/笛卡尔积
  - 关系运算符：选择/投影/连接/除
- 关系演算

关系型数据库的使用

- SQL语言
  - 数据查询（DQL）
  - 数据定义（DDL）
  - 数据增删改（DML）
  - 数据访问控制（DCL）

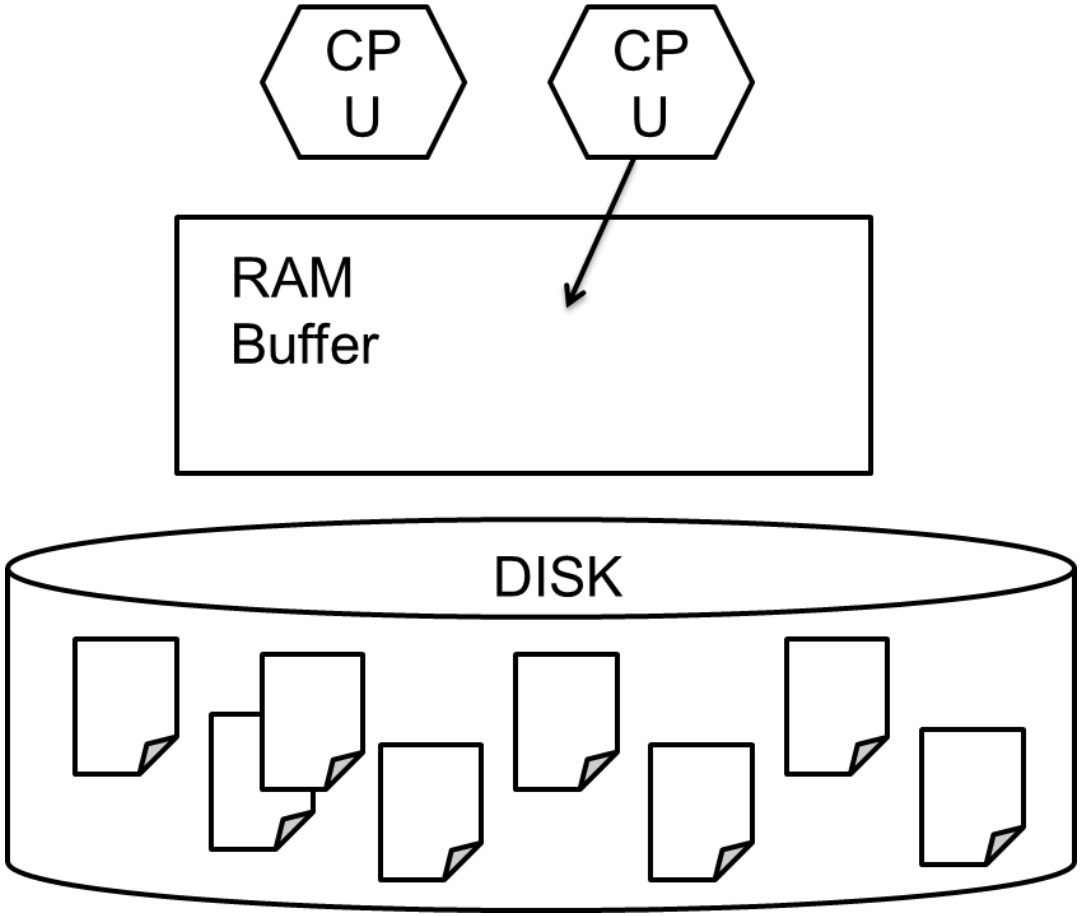
https://www.runoob.com/sql/sql-tutorial.html

关系型数据库的使用

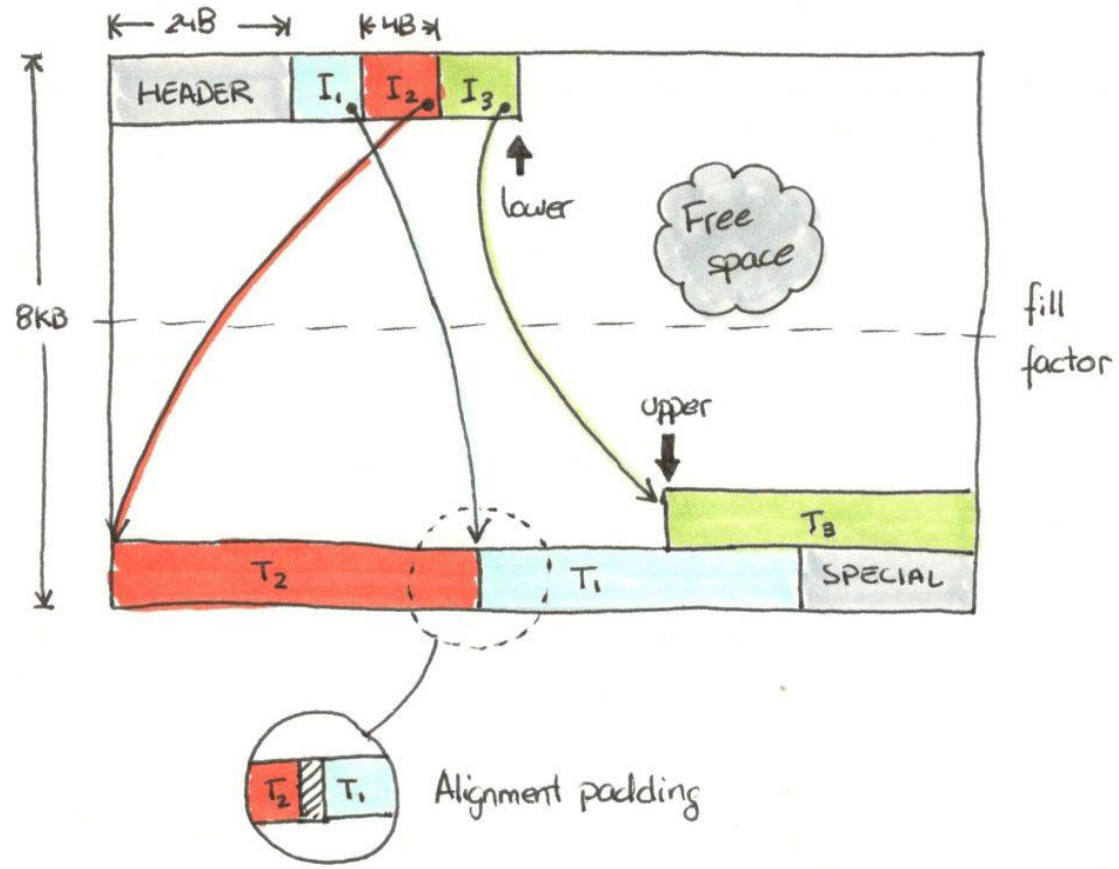
- 应用程序访问数据库
  - ODBC：Open Database Connectivity
  - JDBC：Java based Database Connectivity
  - Native APIs

https://www.runoob.com/sql/sql-tutorial.html

关系型数据库的存储



关系型数据库的存储

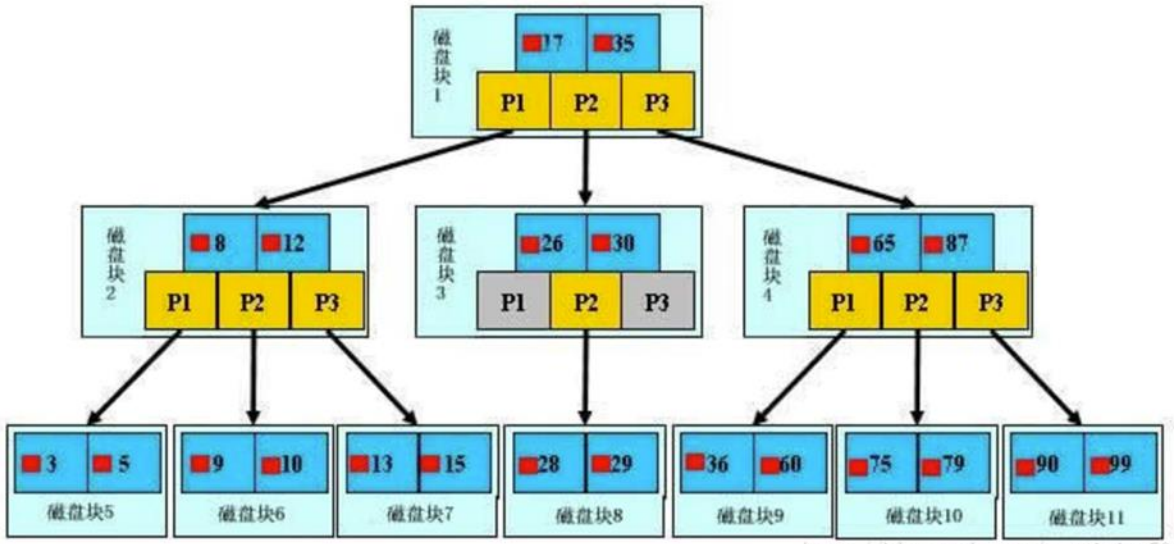

2B
4B
HEADER
I1
I2
I3
lower
Free space
8KB
fill factor
upper
T3
T2
T1
SPECIAL
T2
T1
Alignment padding

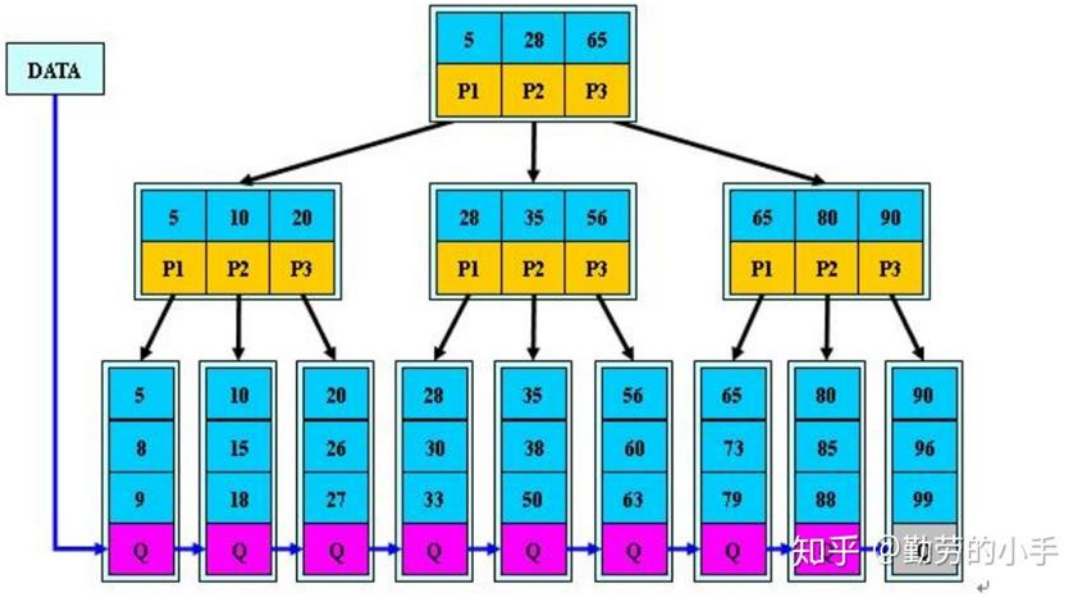
关系型数据库的查询

- 查询索引
  - B-Tree
  - Hash
  - Skip-list
  - Inverted Index
  - R-Tree
  - Z-order Curve

关系型数据库的查询

- 查询索引
  - B树/B+树





关系型数据库的查询

- SQL → Plans： Interpretation
- Plans → Best Plan： Query Optimization
- Best Plan → Results： Query Evaluation

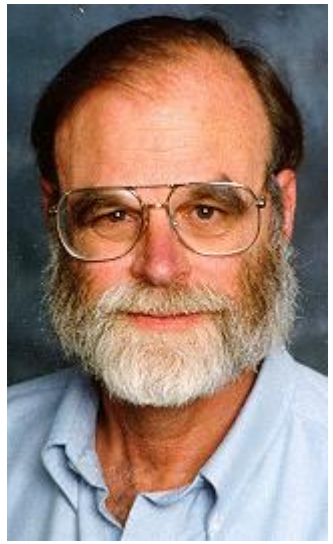
关系型数据库的查询

- 单个操作执行过程
  - 选择 Selection
  - 映射 Projection
  - 链接 Join
  - 排序 Sort
  - 聚集（分组聚集） Aggregation
  - …

关系型数据库的查询

- 查询优化
  - 时间估计
    - 简单计算/直方图
  - 查询改写
    - 去除Distinct
    - 子查询改连接
    - 尽量不用中间表
    - 相关子查询优化
    - …

关系型数据库的事务

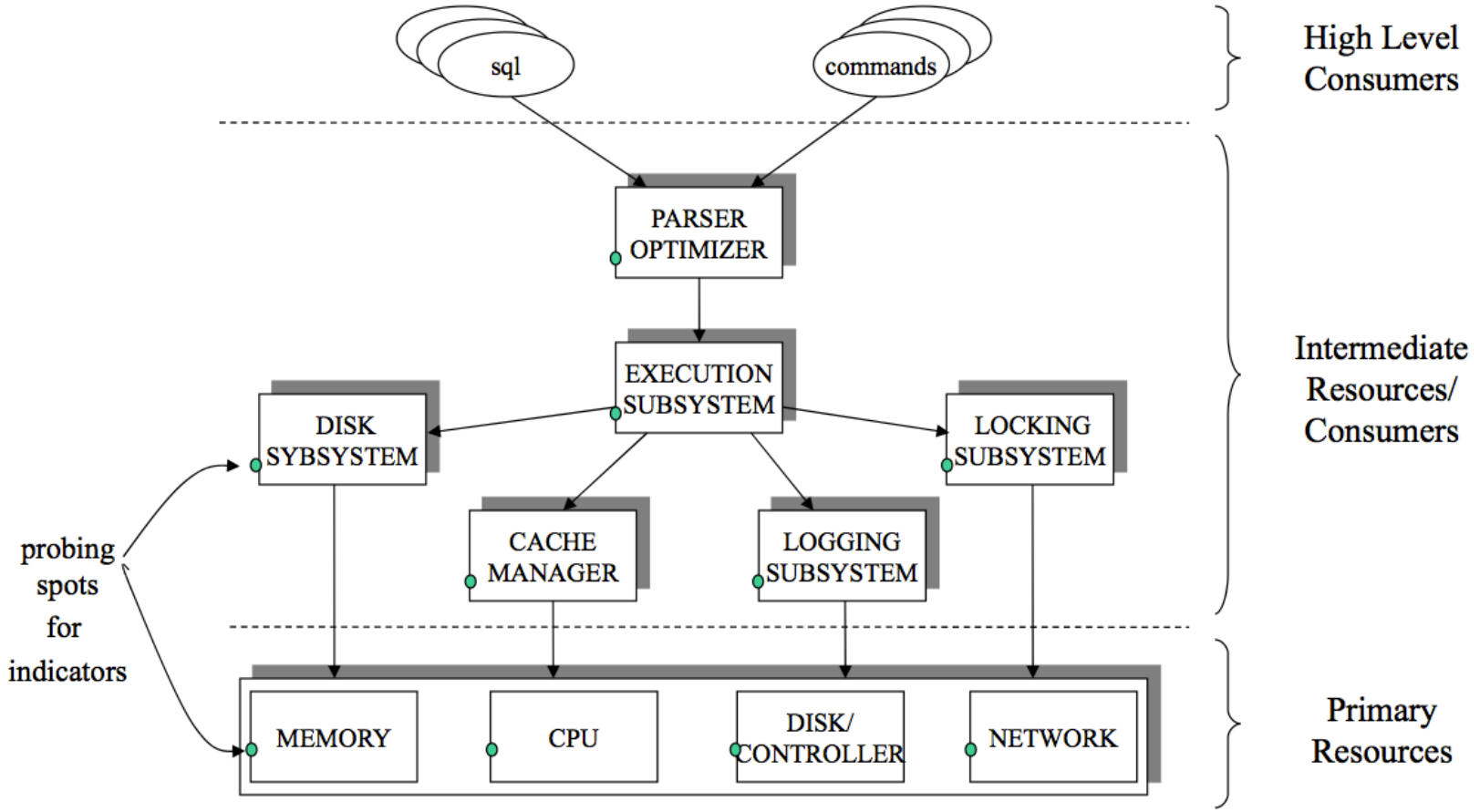


- **原子性（Atomicity）**
  - 一个事务（transaction）要么没有开始，要么全部完成，不存在中间状态。
- **一致性（Consistency）**
  - 事务的执行不会破坏数据的正确性，即符合约束。
- **隔离性（Isolation）**
  - 多个事务不会相互破坏。
- **持久性（Durability）**
  - 事务一旦提交成功，对数据的修改不会丢失。

关系型数据库的事务

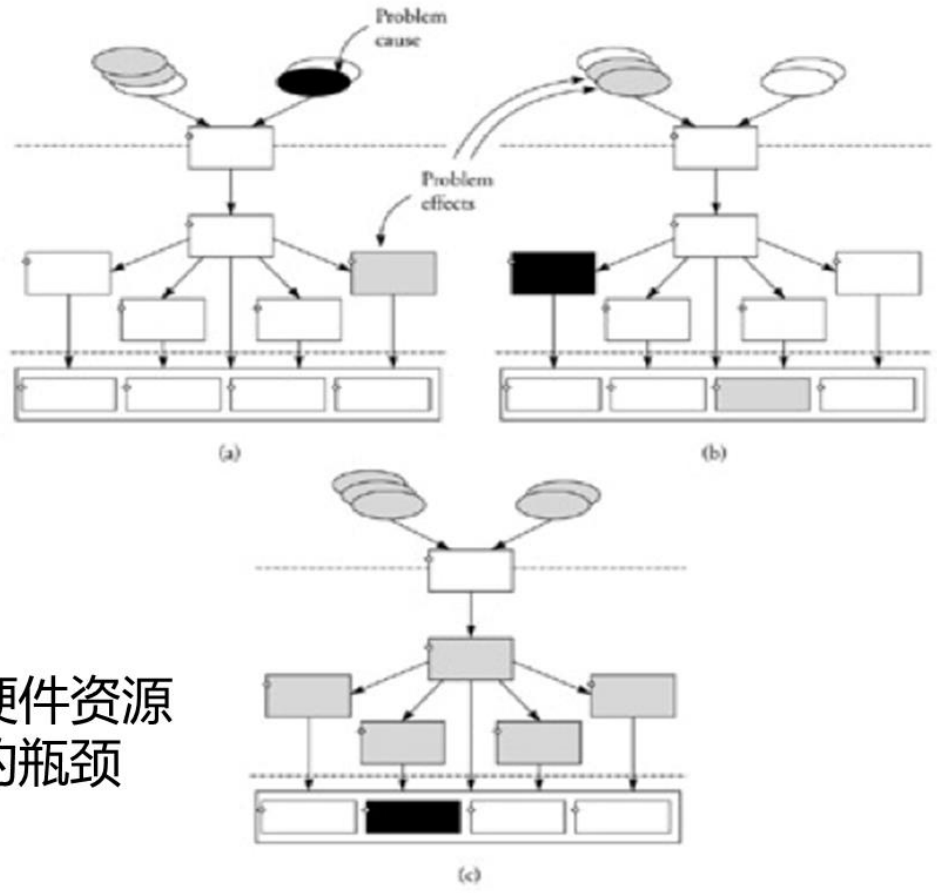
- **原子性：日志**
  - Undo/Redo/checkpoint
- **一致性**
  - 一致性策略/一致性检查
- **隔离性：并发控制**
  - 锁/多版本/验证机制
  - 可串行化调度/冲突
- **持久性：日志**
  - Undo/Redo/checkpoint

关系型数据库的调优


sql
commands
High Level Consumers
PARSER OPTIMIZER
EXECUTION SUBSYSTEM
DISK SYBSYSTEM
LOCKING SUBSYSTEM
CACHE MANAGER
LOGGING SUBSYSTEM
Intermediate Resources/ Consumers
probing spots for indicators
MEMORY
CPU
DISK/ CONTROLLER
NETWORK
Primary Resources

关系型数据库的调优

应用程序构造或查询书写的问题



数据库组件的瓶颈

硬件资源的瓶颈

关系型数据库的设计

- **逻辑设计**
  - 实体
  - 关系
  - ER图
  - 范式化:第1/2/BC/4范式
- **物理设计**
  - 各种优化： 冗余属性/推导属性/表分裂/物化视图/存储过程/触发器等

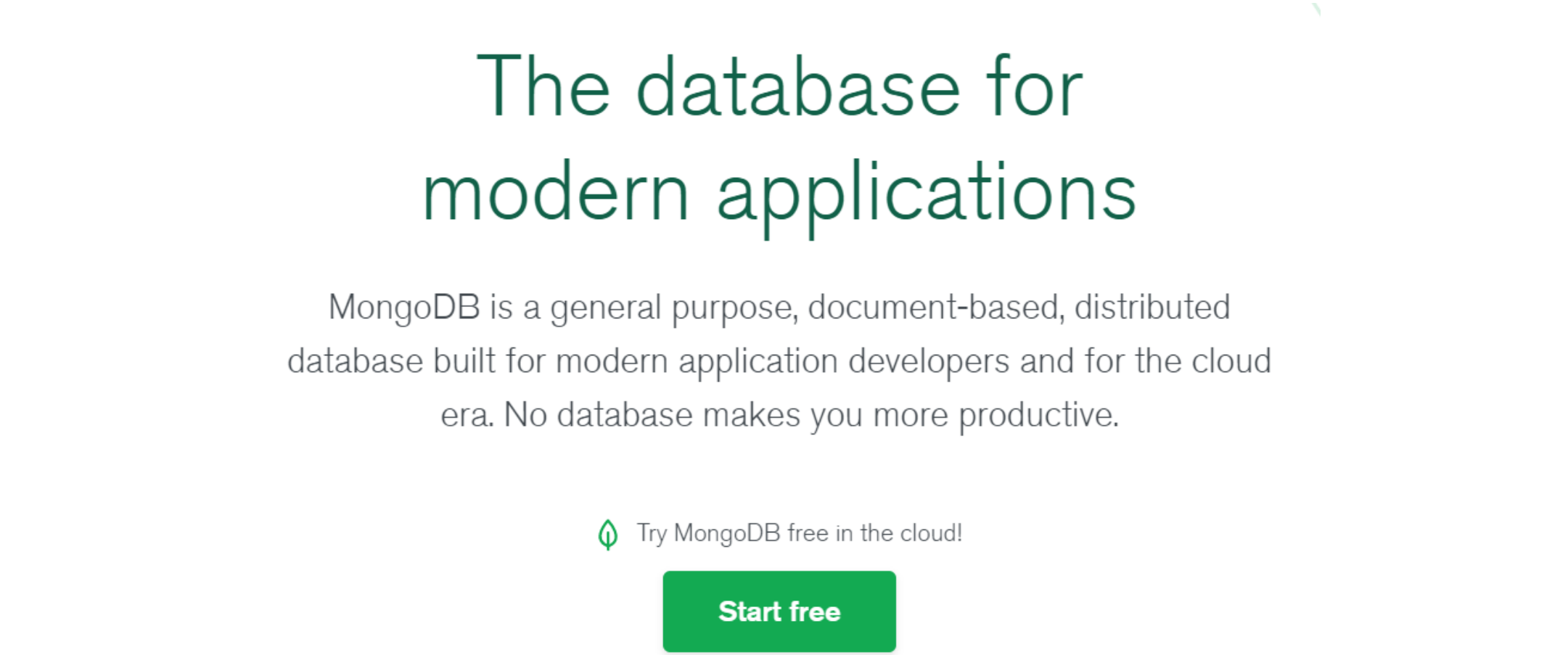
提纲：SQL/NoSQL/NewSQL

- 数据库的设计思想
  - 数据库为什么长这样
- 数据库的使用
  - 增删改查
- 重要的内部实现细节
  - 存储/查询/事务/可用性…
- 设计练习

关系型数据库的缺点

- 只能存储为表格？非规则数据如文本等怎么存储？

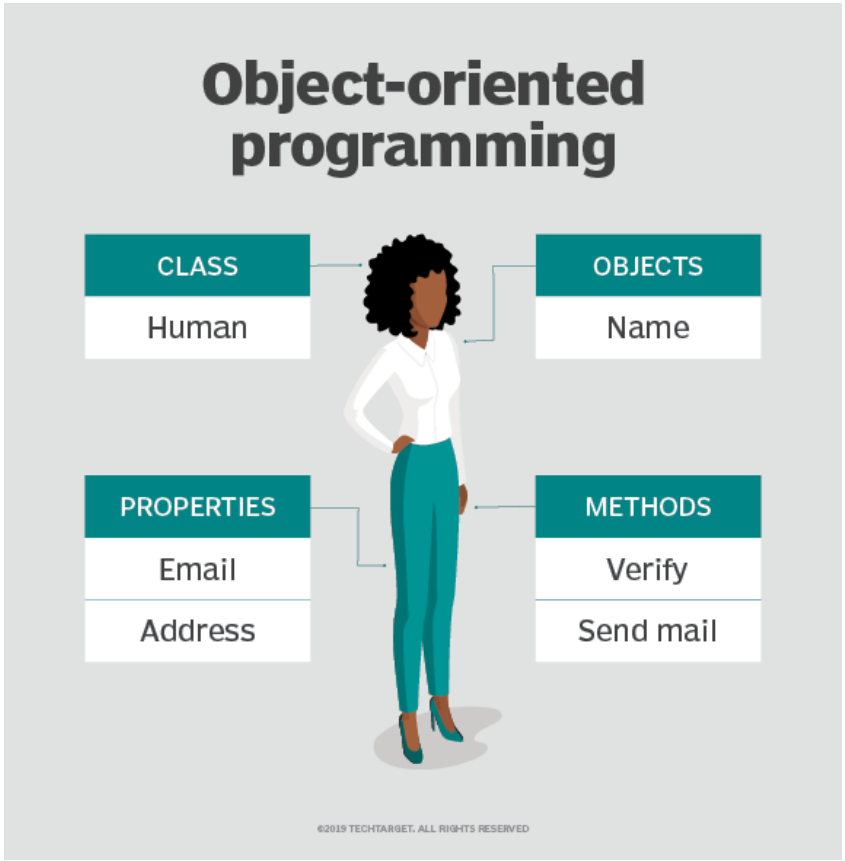
文档型数据库



https://www.mongodb.com/

文档型数据库的设计思想

- 面向对象

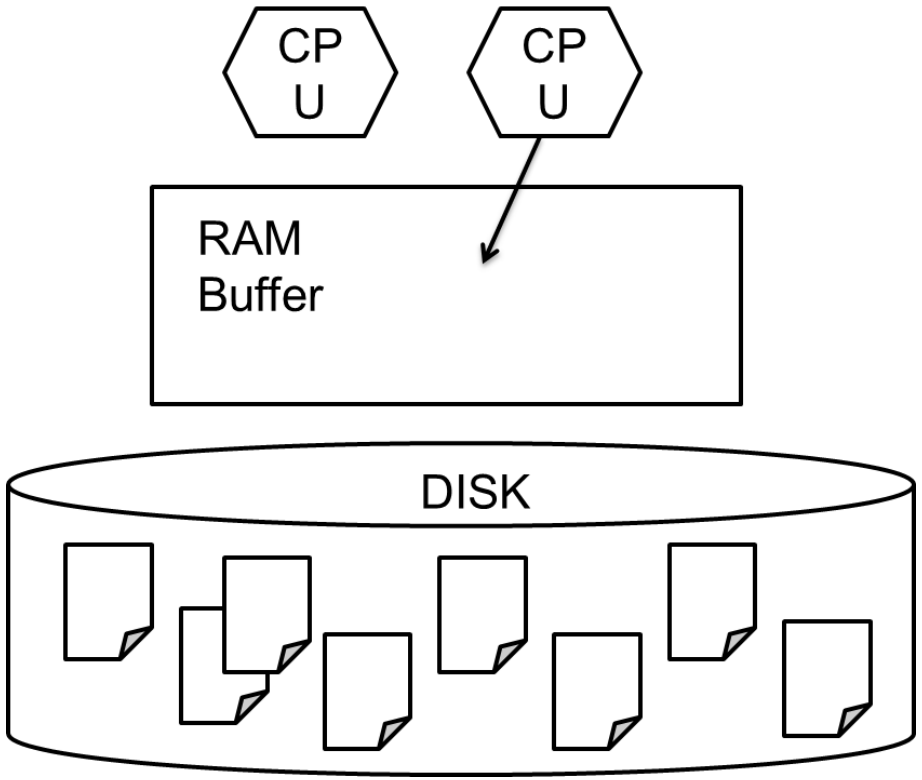


文档型数据库的使用

- 增删改查：
  - https://docs.mongodb.com/manual/

文档型数据库的存储

- 如何存储?



文档型数据库的存储

- **如何保证存储的正确性?**
  - 人为因素：删库跑路、程序错误
  - 系统因素：介质损坏、并发控制

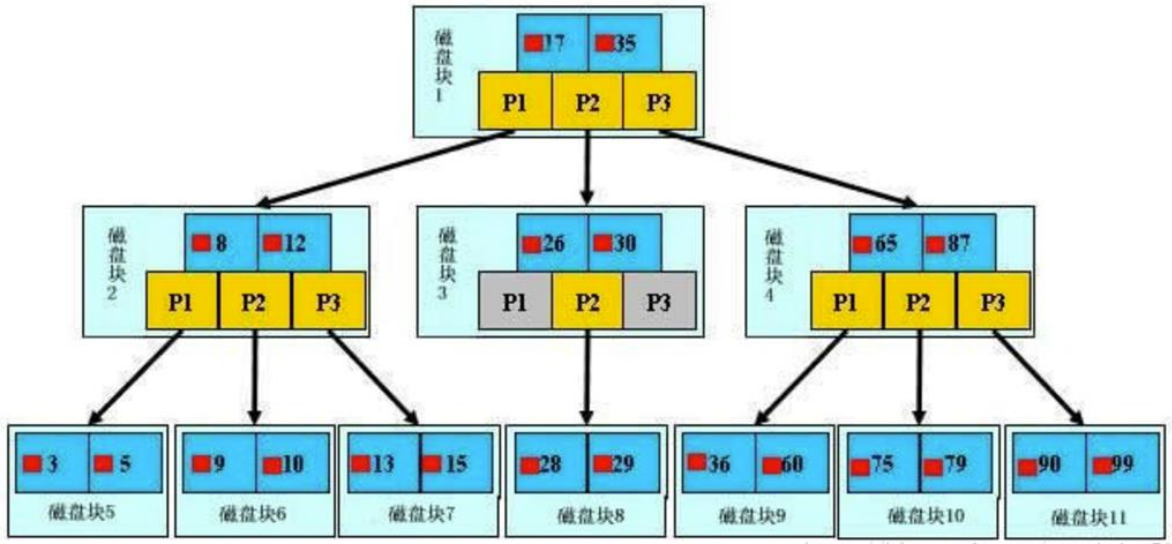
    →原子性的两重含义
- **如何衡量存储的正确性?**
  - 持久性（Durability）

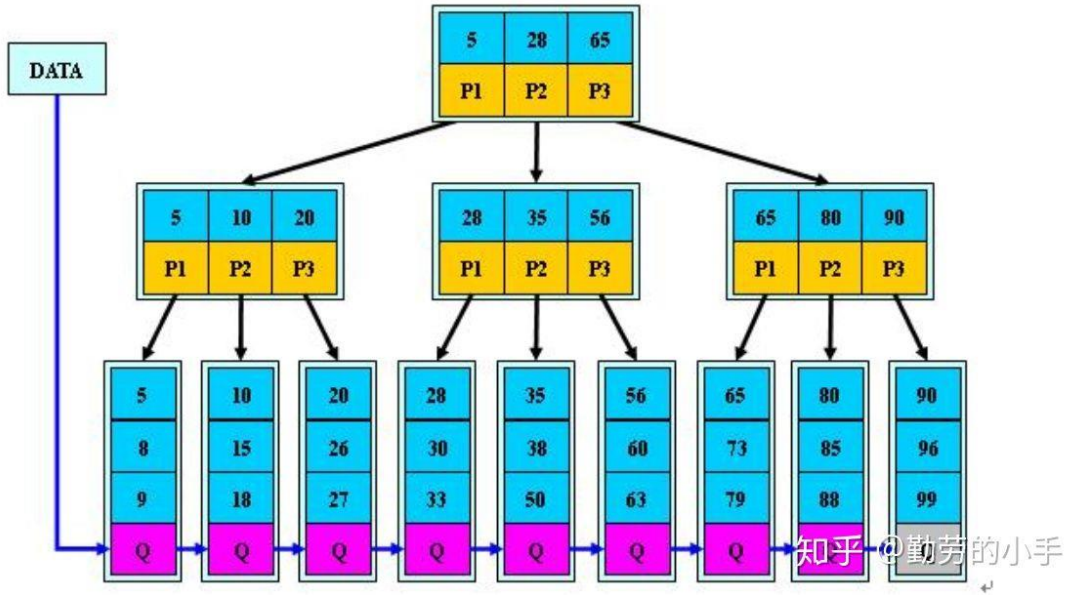
文档型数据库的存储

- **原子性的两重含义**
  - 不受宕机影响：undo/redo日志/Checkpoint
  - 进行并发控制：可线性化/加锁/多版本

文档型数据库的查询

- B树/B+树





文档型数据库的事务

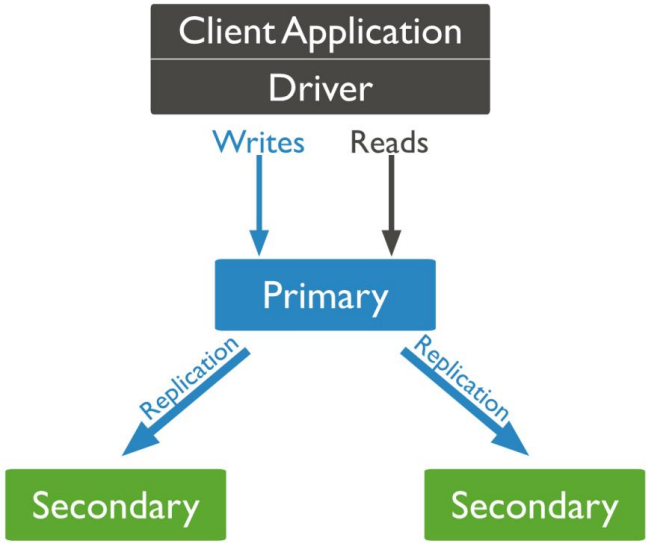
- NoSQL数据库通常不支持事务!
  - 只保证单个操作的原子性
  - MongoDB的Multi-document Transaction只限于单节点
- 原因：事务的性能得不到保证
- 把数据一致性的问题交给应用来解决

文档型数据库的事务

- 文档型数据库如何解决数据一致性问题?
  - 一块嵌入
  - 同步标记
  - 消息队列
  - 日志/时间戳
  - 操作系统原子操作：锁/信号量/CAS

文档型数据库的可用性

- 高可用与共识机制
  - 至少一主两备
  - Raft协议（http://thesecretlivesofdata.com/raft/）



https://raft.github.io/

文档数据库的设计

- 对象
- 对象之间的关系（M:N）
- 范式化与逆范式化

提纲：SQL/NoSQL/NewSQL

- 数据库的设计思想
  - 数据库为什么长这样
- 数据库的使用
  - 增删改查
- 重要的内部实现细节
  - 存储/查询/事务/可用性…
- 设计练习

问题1

- 数据模型的设计应该以OO应用程序端为主，还是以数据库端为主?
- 先设计数据库还是先设计程序架构?

问题1

- 观点一：业务流程为主，因此程序架构为主
- 观点二：State of the World为主，因此数据库为主

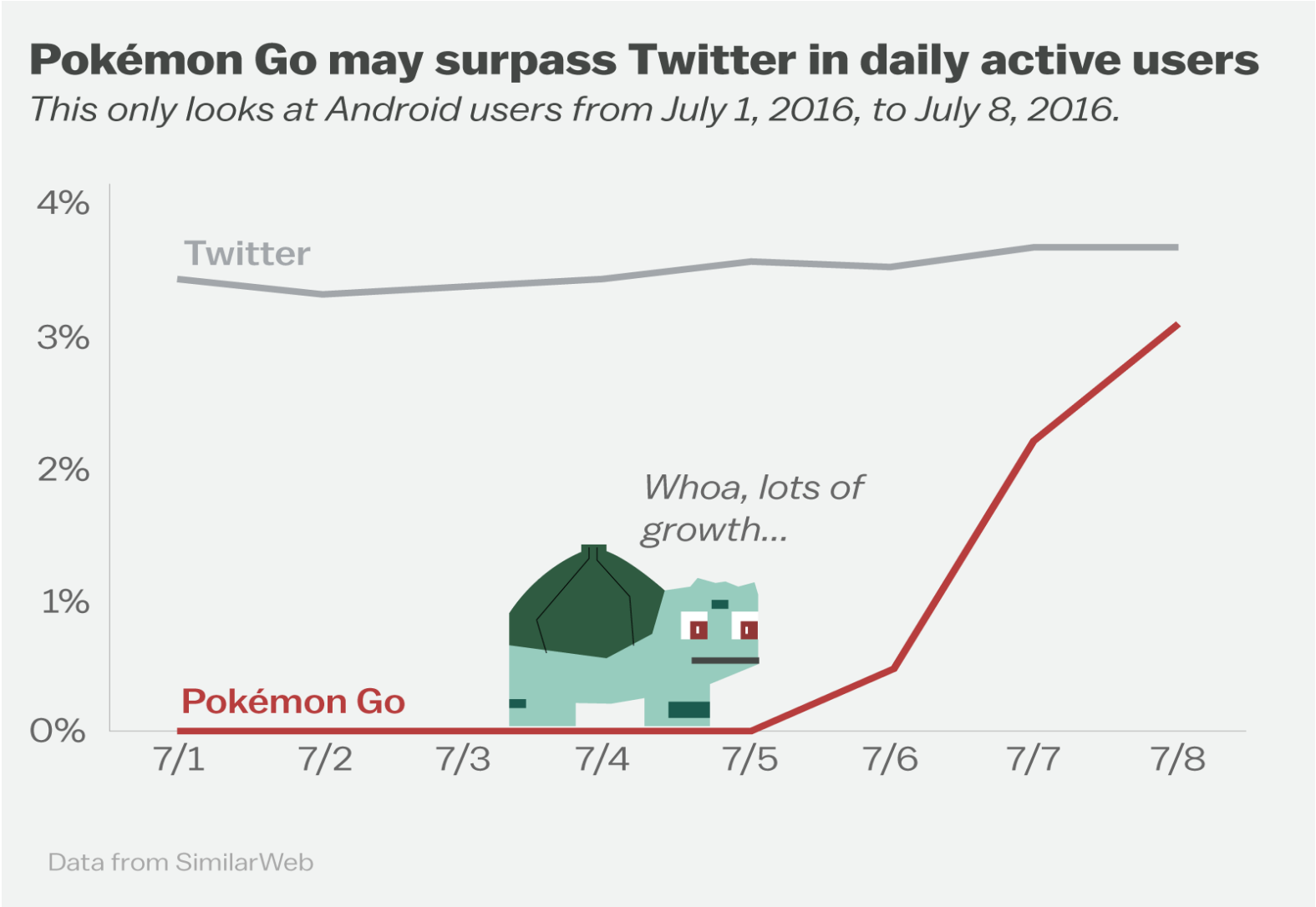
问题1

- 观点一场景
  - 简单功能的数据库
  - 简单模型的数据库
  - 传统的数据库设计过程不再适用
    - ERD → ORM
  - NoSQL适用场景更广

问题1

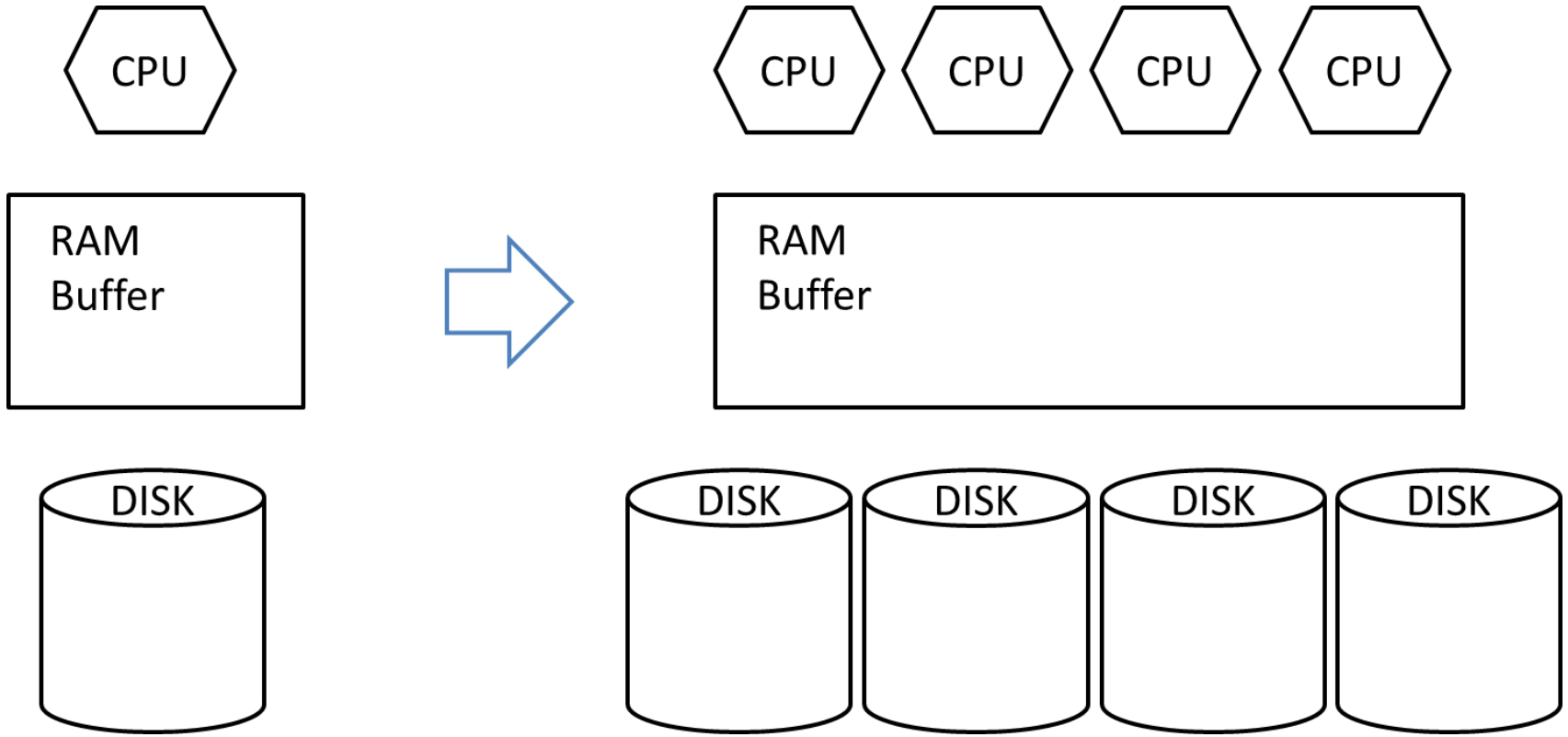
- 观点二场景
  - 传统SQL数据库
  - 需要进一步优化SQL数据库的性能，拓展其功能
    - ORDBMS
    - NewSQL
    - HTAP

问题2


Pokémon Go may surpass Twitter in daily active users
This only looks at Android users from July 1, 2016, to July 8, 2016.
4%
3%
2%
1%
0%
Twitter
Whoa, lots of growth...
Pokémon Go
7/1
7/2
7/3
7/4
7/5
7/6
7/7
7/8
Data from SimilarWeb

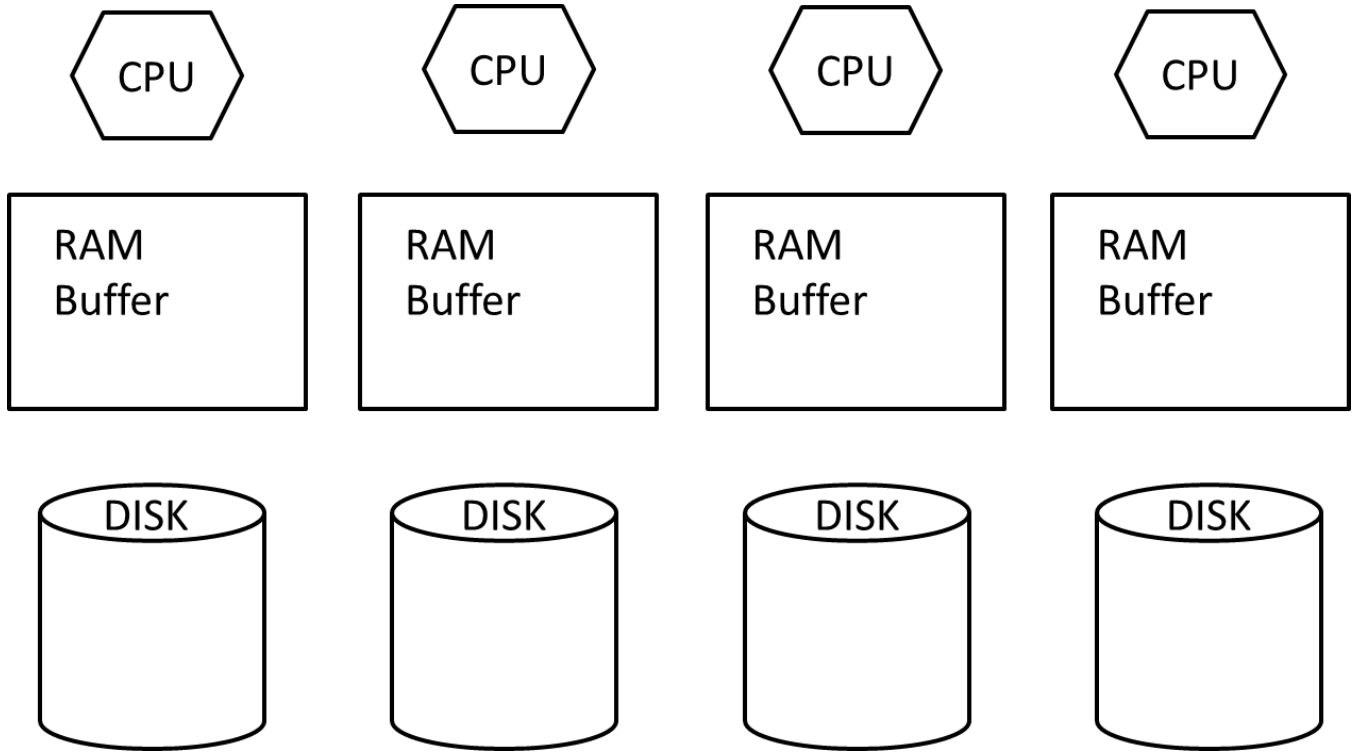
问题2

- Scale Up

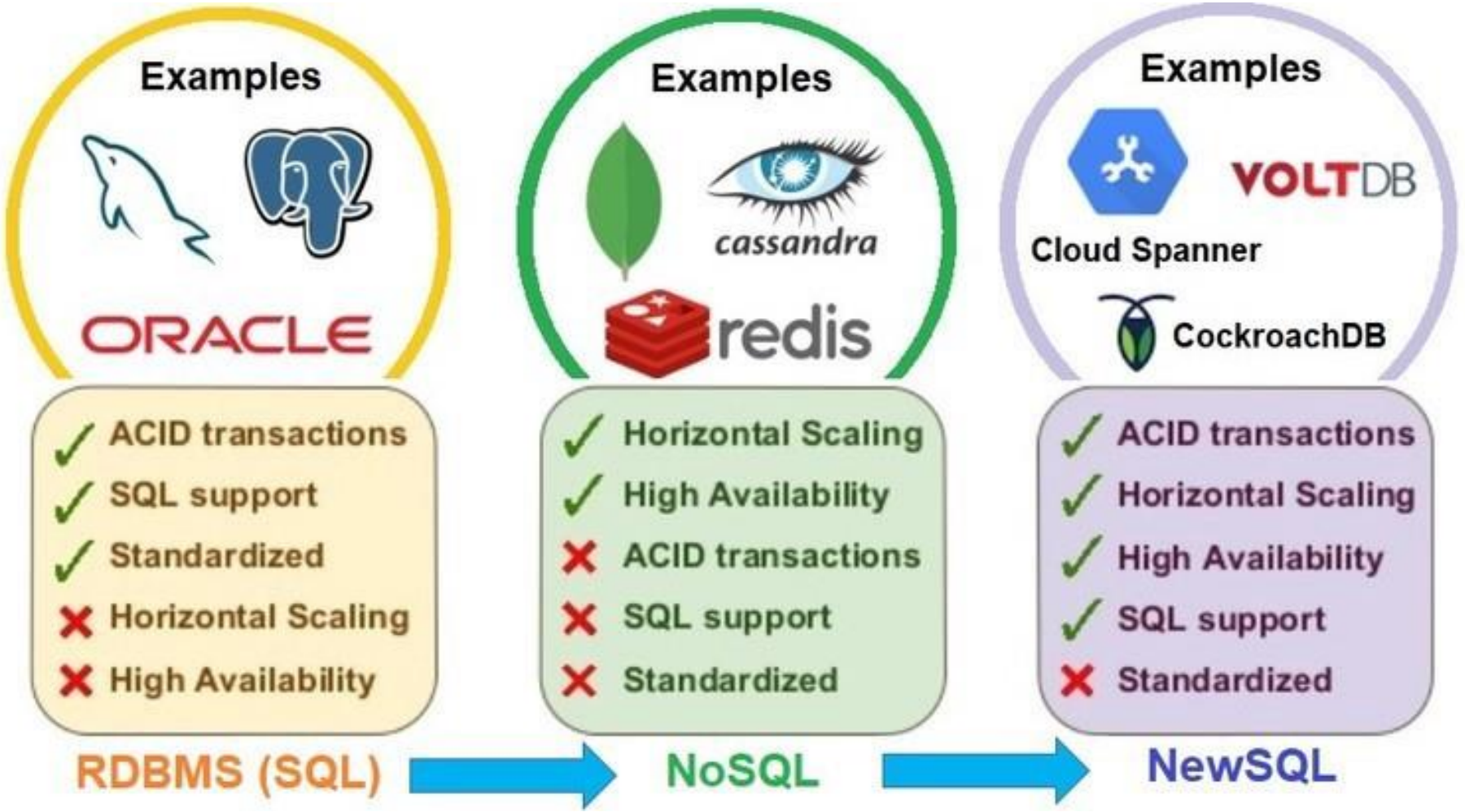

CPU
RAM
Buffer
DISK
CPU
CPU
CPU
CPU
RAM
Buffer
DISK
DISK
DISK
DISK

问题2

- Scale Out



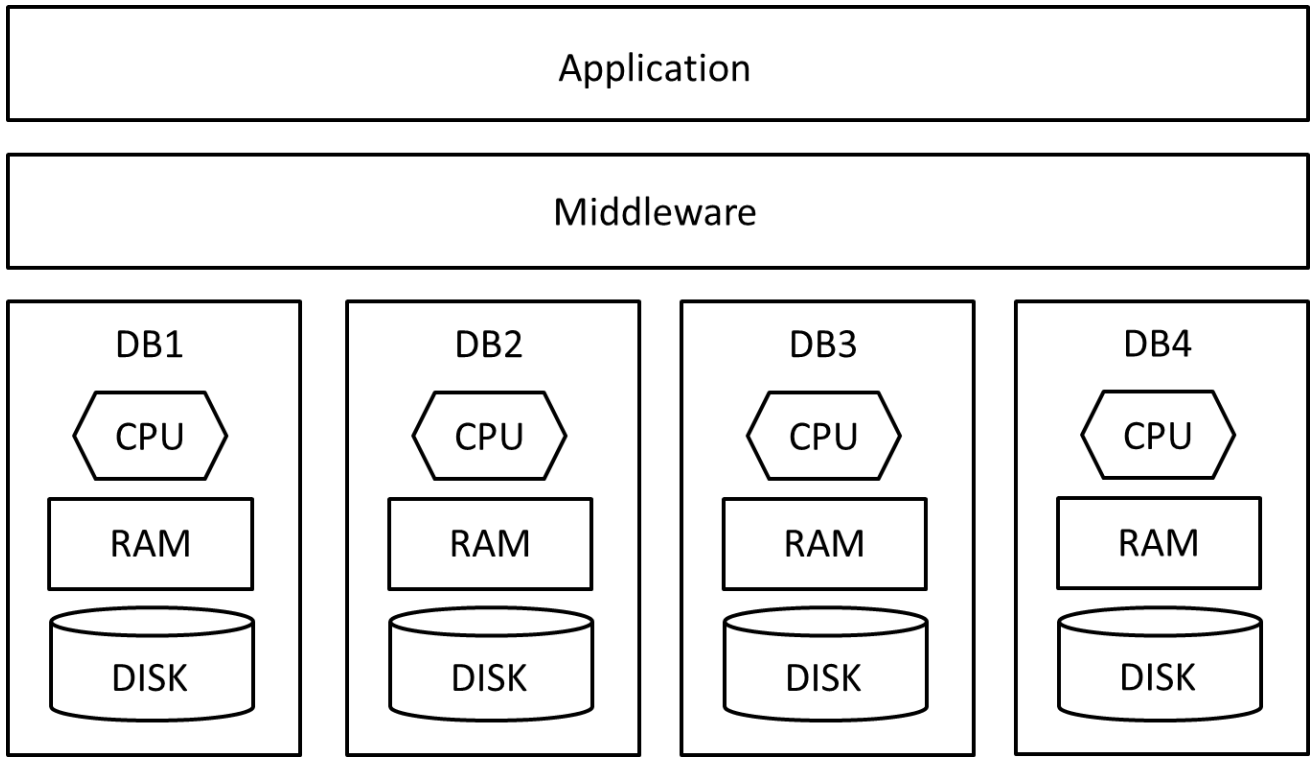
NewSQL


Examples
ORACLE
✓ ACID transactions
✓ SQL support
✓ Standardized
✗ Horizontal Scaling
✗ High Availability
RDBMS (SQL)
Examples
cassandra
redis
✓ Horizontal Scaling
✓ High Availability
✗ ACID transactions
✗ SQL support
✗ Standardized
NoSQL
Examples
VOLTDB
Cloud Spanner
CockroachDB
✓ ACID transactions
✓ Horizontal Scaling
✓ High Availability
✓ SQL support
✗ Standardized
NewSQL

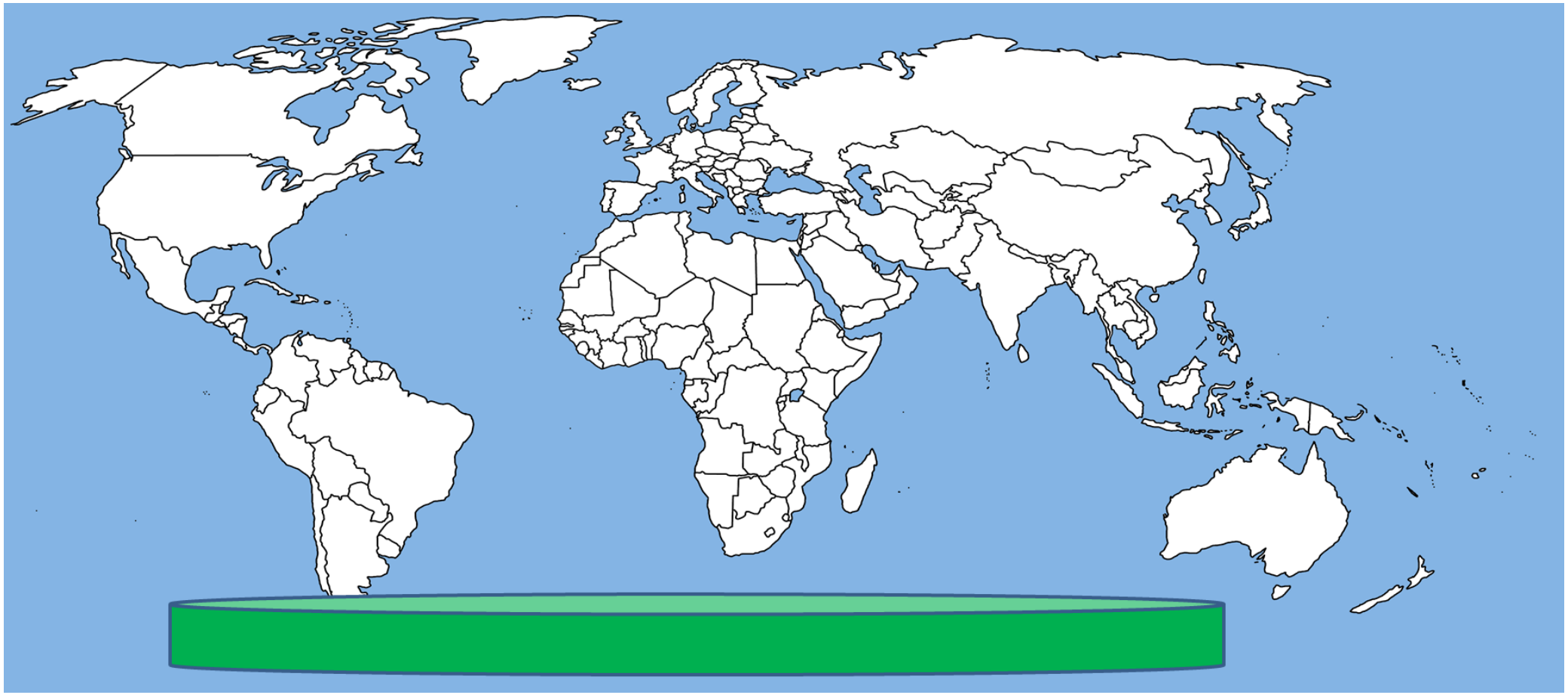
How to Scale Out?

How to Scale Out?

- 分库/分表

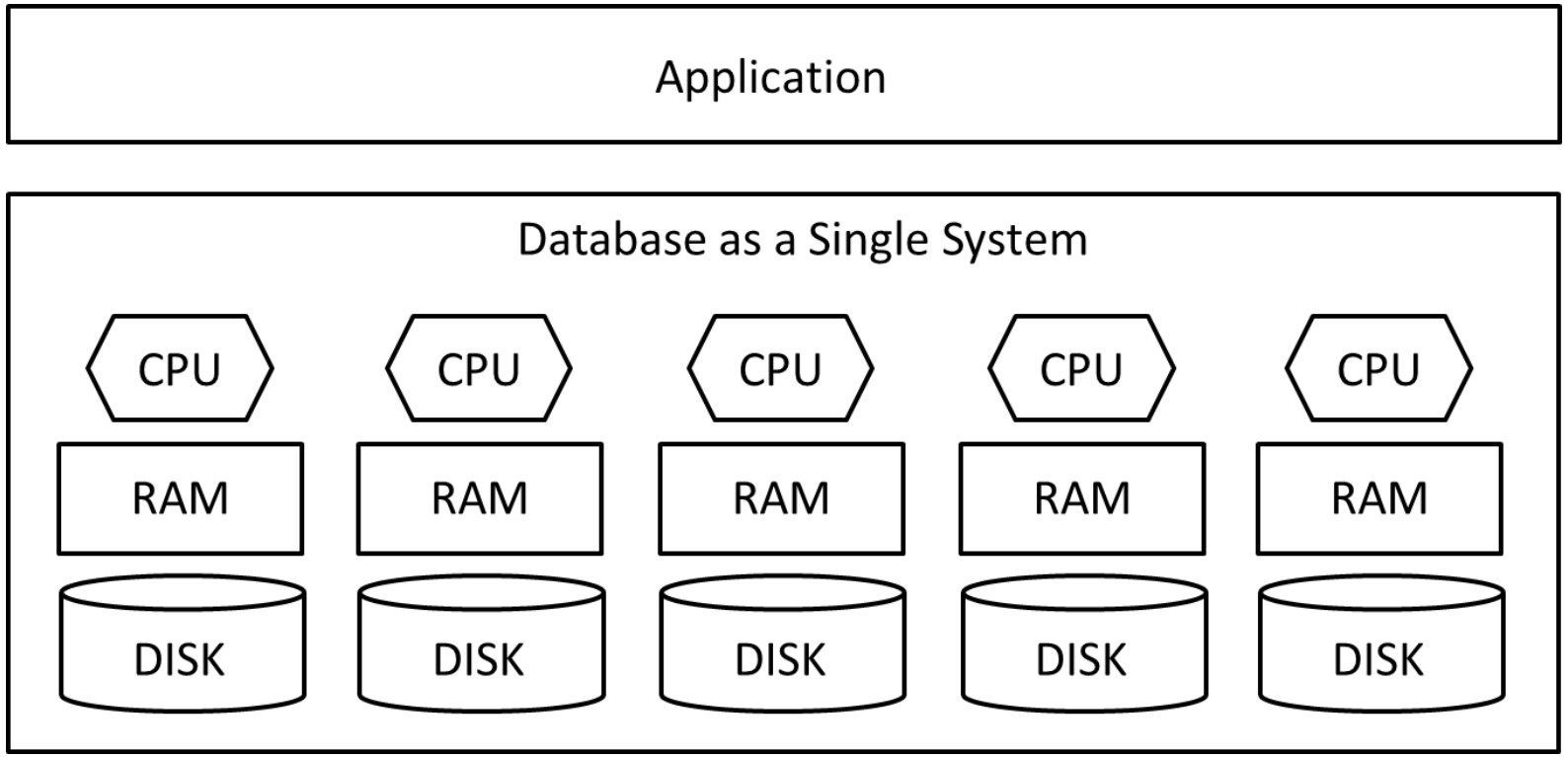


How to Scale Out?



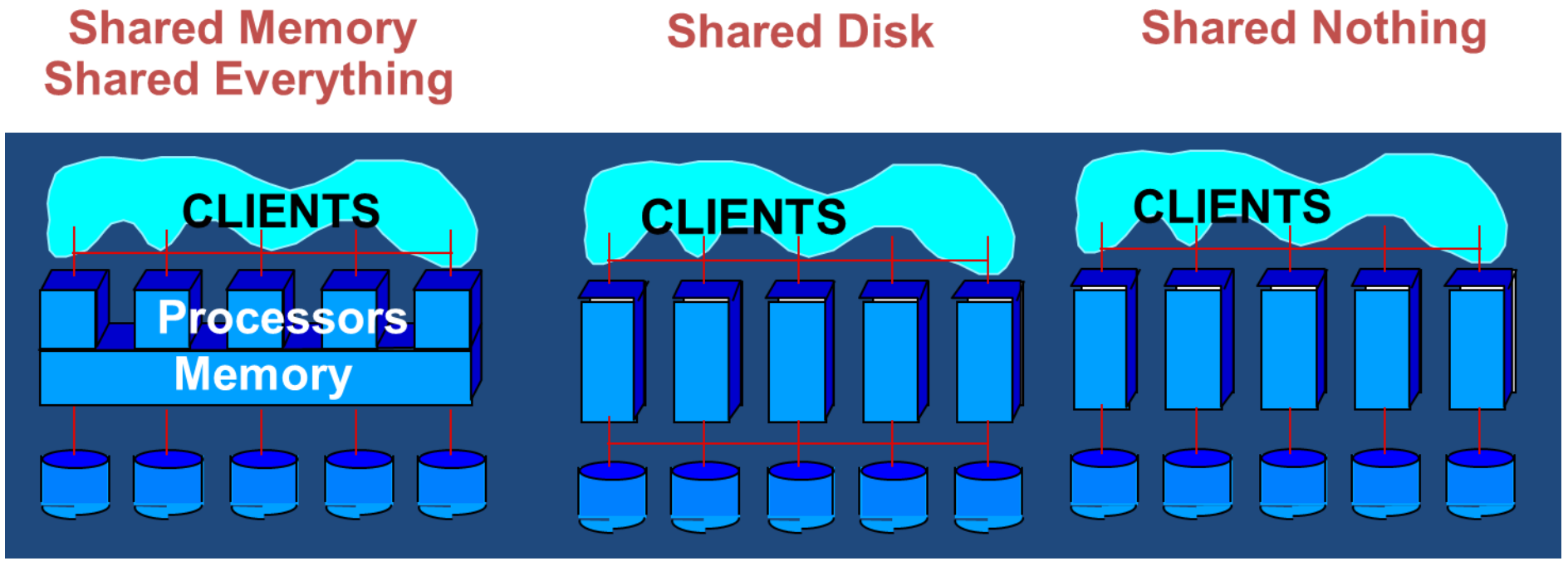
How to Scale Out?

- 并行/分布式数据库



How to Scale Out?

- 并行/分布式数据库



How to Scale Out?

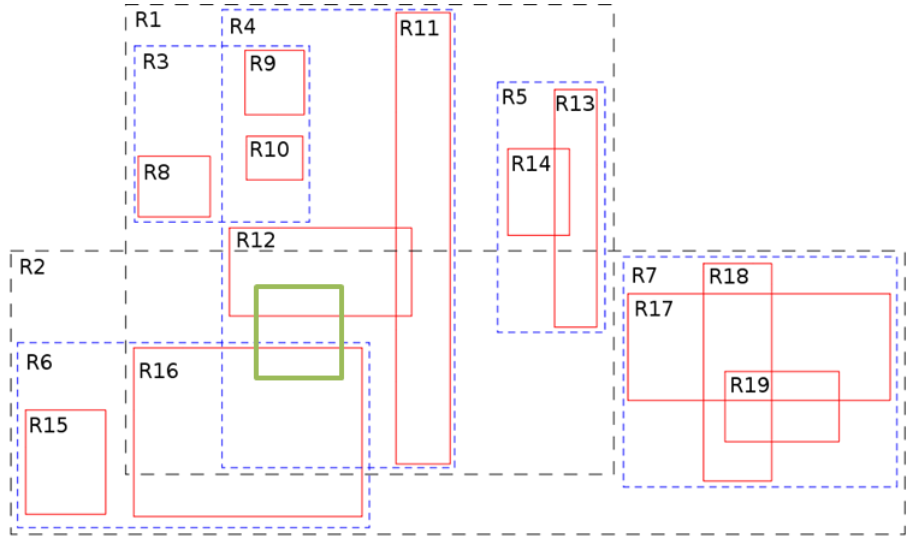
- 并行/分布式数据库
  - 如何划分数据?
    - Round robin
    - Hash partitioning
    - Range partitioning
  - 如何执行查询?
    - 关系代数的并行化

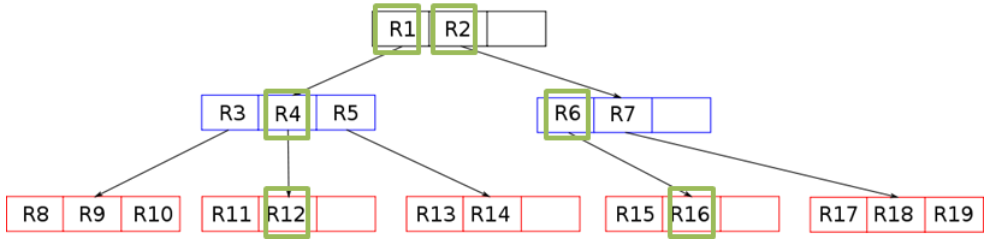
分布式数据库使用

- https://pingcap.com/docs-cn/

分布式数据库的查询

- 索引举例： R-Tree Search



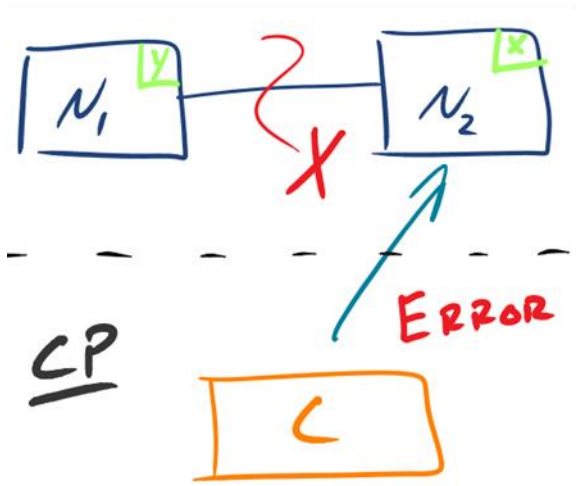


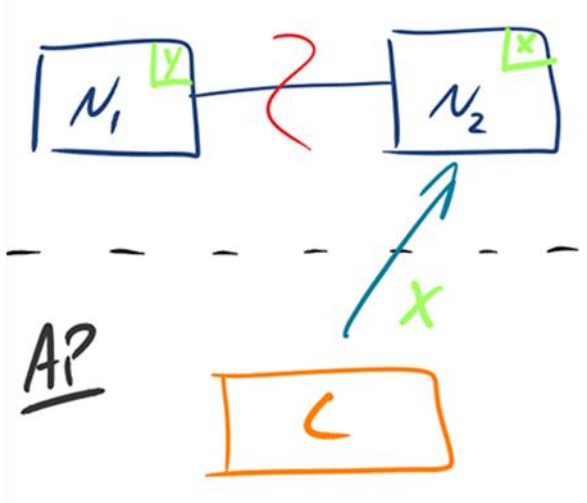
分布式数据库的事务

- A
  - 日志/2PC
- C
- I
  - 分布式锁/多版本与时间戳
- D
  - 日志/2PC

分布式数据库的高可用

- P必须有，AP/CP二选一





分布式数据库的高可用

- **Node Availability vs Service Availability**
  - Node Availability (nA)
  - Service Availability (sA)

- **CnAP vs CsAP**
  - CnAP：只能在CP和nAP中二选一。
  - CsAP：某种情况下可以兼顾

分布式数据的高可用

- **CAP理论**
  - 一致性（Consistency）
  - 可用性（Availability）
  - 分区容错性（Partitioning Tolerance）

  - 任意一个分布式数据库只能在C、A和P三者中兼顾两个。

分布式数据库的备份

- Backup vs Replication
  - 预防错误或灾难
  - 可以处理更多错误，例如误操作（已提交的事务需要撤销）
  - 利用Replication机制可实现更精确的恢复（例如，利用日志恢复到某个事务节点）

分布式数据库的备份

- Backup vs Replication
  - 实现高可用
  - 在一定程度上也可以起到Backup的作用，但不完全
  - 无法撤销已提交的事务（检查点之前的日志可能被删除）

分布式数据库的备份

- 数据存档（Archiving）
  - 磁带
  - 磁盘
  - …

课程小结

	OldSQL	NoSQL	NewSQL
Data model	Relational	---	Relational
Interface	SQL	Variance	SQL
Consistency/Concurrency control	Strong	Weak	Strong
Fault tolerance	Strong	Fine	Strong
Performance	Poor	Good	Good
Scalability	Poor	Good	Fine

补充：数据库应用之OLAP

- 二者的数据有何区别？
  - OLTP： State of the Current World
  - OLAP： History
- 二者的负载有何区别？
  - OLTP： Query & Update
  - OLAP： Aggregation Query

补充：数据库应用之OLAP

- 数据仓库的数据模型
  - Data Cube
- 数据仓库系统
  - MOLAP vs ROLAP
- 数据仓库的数据模式
  - 雪花模型
- 现代语境下的数据分析
  - 数据湖/AI…

补充：数据库应用之搜索引擎

- **倒排索引**
- **遇到的问题**
  - 内存无法容下所有的索引?
    - Blocked sort-based Indexing
  - 单个计算机无法容下所有的索引?
    - 分布式索引
  - 索引项还在增加?
    - 静态索引+动态索引

补充：The Big Picture

- 数据存储
  - Files, HDFS, DBs
- 数据移动
  - Messaging systems: Kafka, ActiveMQ
  - ETL tools: sqoop, Flume, FiNi
- 数据处理
  - Python, DBs, Spark, Flink
- 分析工具
  - OLAP, R, Mahout, SAS
- 机器学习
  - Tensorflow/PyTorch

Thanks!