第2章 关系数据模型与SQL的发展

大数据管理概论

第2章 关系数据模型与SQL

- **教学内容**
  - 本章讲述关系数据模型与SQL语言的基础知识与概念；
  - 面向大数据管理需求的数据库实现技术及SQL扩展技术；
  - 通过代表性数据库分析介绍关系数据库的主要实现技术。
- **教学目标**
  1）能够阐述关系数据库的基本概念、操作与实现技术；
  2）能够阐释SQL基本语法和扩展语法的特征；
  3）能够描述SQL on Hadoop的典型应用案例；
  4）能够列举并陈述NoSQL数据库的特点；
  5）能够列举并陈述代表性的关系、MPP、NewSQL数据库。

第2章 关系数据模型与SQL

2.1 关系数据库概述

2.2 关系数据库标准语言SQL

- SQL for XML
- SQL for JSON

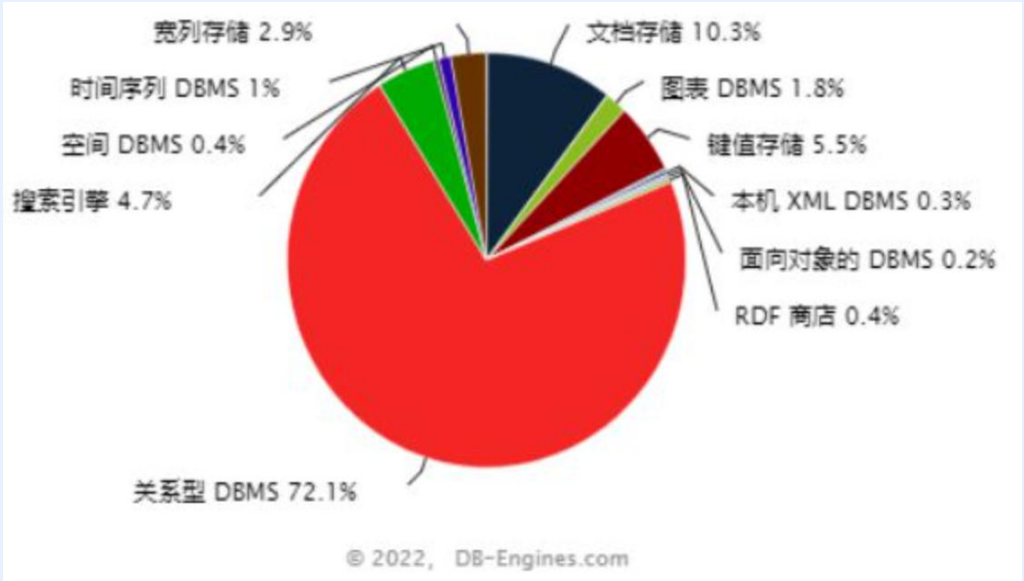
2.3 SQL on Hadoop

2.4 NoSQL数据库 vs. 关系数据库

2.5 代表性数据库演化与发展趋势

2.4 NoSQL数据库

- **NoSQL数据库**：常指分布式存储的非关系型数据库，主要应用于不适合关系存储的KV数据库、列存储数据库、文档数据库、图数据库、时序数据库等应用领域。
- **主要特点**：
  - **无模式设计**——用户可随时自定义数据存储格式并在运行中修改数据格式。
  - **弱一致性**——通常采用最终一致性（eventual consistency），配合多副本机制。
  - **易扩展性**——常采用SN(shared-nothing)结构的分布式存储，可使用大规模廉价服务器集群，**通常不支持连接操作**，数据水平分布，嵌套方式存储。
  - **高并发读写性能**——结构简单，**弱化ACID，强化高并发读写**性能。



2.4 NoSQL数据库

- **四大NoSQL数据库——键值（key-value）数据库**

  Dynamo, Memcached, Redis, SimpleDB...

数据模型	• 键/值对 • 键是一个字符串对象 • 值可以是任意类型的数据，比如整型、字符型、数组、列表、集合等
典型应用	• 涉及频繁读写、拥有简单数据模型的应用 • 内容缓存，比如会话、配置文件、参数、购物车等； • 存储配置和用户数据信息的移动应用
优点	扩展性好，灵活性好，大量写操作时性能高
缺点	无法存储结构化信息，条件查询效率较低
不适用场景	• 不通过key而是value来查找 • 需要存储数据之间的关系 • 需要事务支持

Key_1	Value_1
Key_2	Value_2
Key_3	Value_1
Key_4	Value_3
Key_5	Value_2
Key_6	Value_1
Key_7	Value_4
Key_8	Value_3

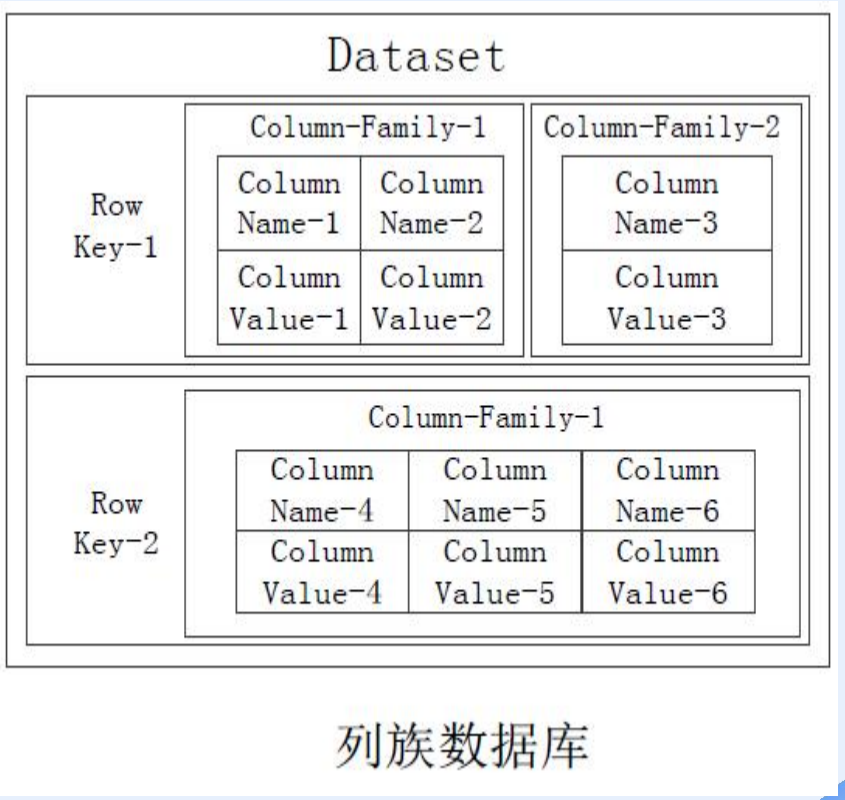
键值数据库

2.4 NoSQL数据库（续）

- **四大NoSQL数据库——列族（column family）数据库**

  Bigtable, HBase, Cassandra, HadoopDB,GreenPlum...

数据模型	KV+列族
典型应用	• 分布式数据存储与管理； • 数据在地理上分布于多个数据中心； • 可以容忍副本中存在短期不一致情况； • 拥有动态字段； • 拥有潜在大量数据，数据量级为TB
优点	• 查找速度快， • 可扩展性强， • 容易进行分布式扩展，复杂性低
缺点	功能较少，大都不支持强事务一致性
不适用场景	需要ACID事务支持



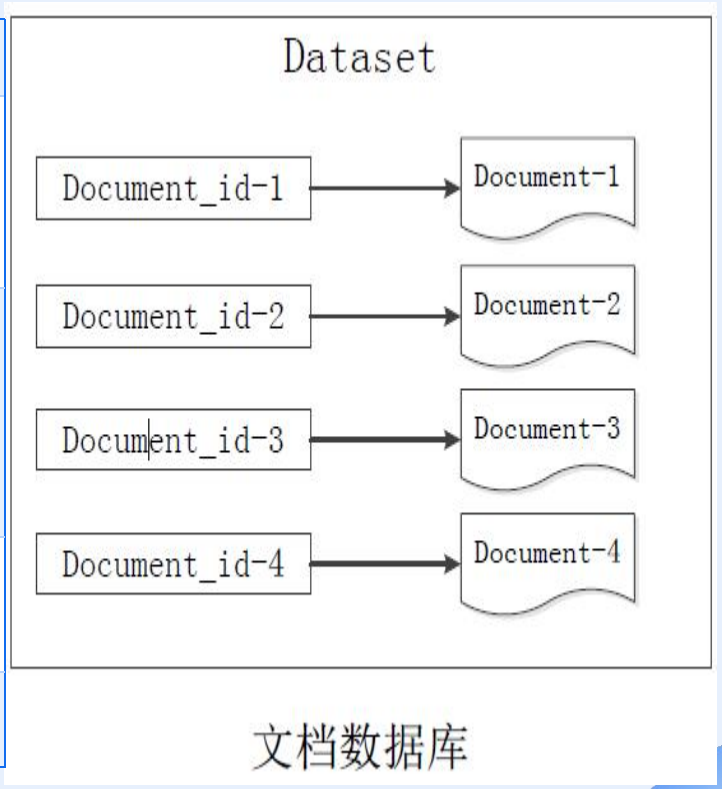
列族数据库

2.4 NoSQL数据库（续）

- **四大NoSQL数据库——文档（document）数据库**

  MongoDB、CouchDB、RavenDB...

数据模型	键/值； 值是版本化的文档
典型应用	• 存储、索引并管理面向文档的数据或者半结构化数据； • 后台具有大量读写操作的网站、使用JSON数据结构的应用、使用嵌套结构等非规范化数据的应用程序
优点	• 性能好(高并发)，灵活性高，复杂性低，数据结构灵活； • 提供嵌入式文档功能，将经常查询的数据存储在同一个文档中； • 既可以根据键来构建索引，也可以根据内容构建索引
缺点	• 缺乏统一的查询语法 • 不支持文档间事务
不适用场景	• 在不同文档上添加事务； 文档DB不支持文档间事务



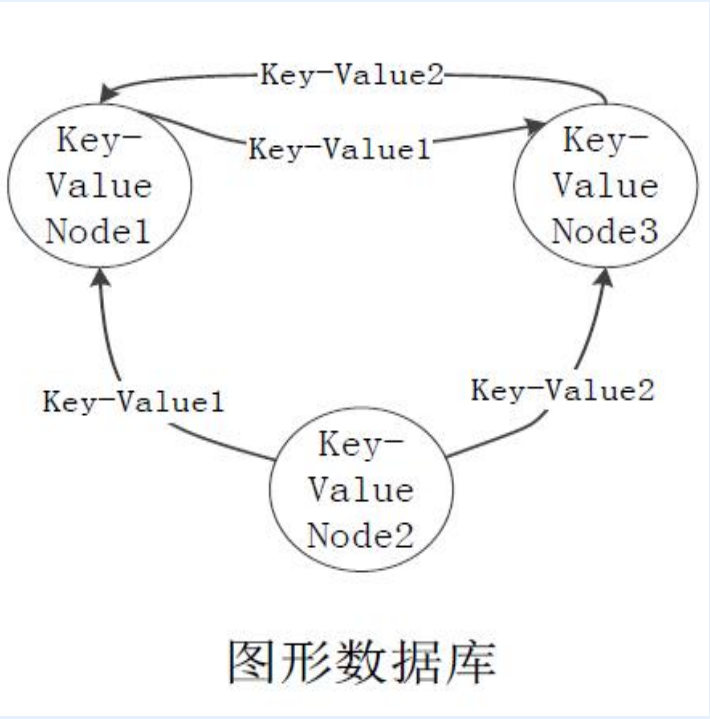
文档数据库

2.4 NoSQL数据库（续）

- **四大NoSQL数据库——图（graph）数据库**

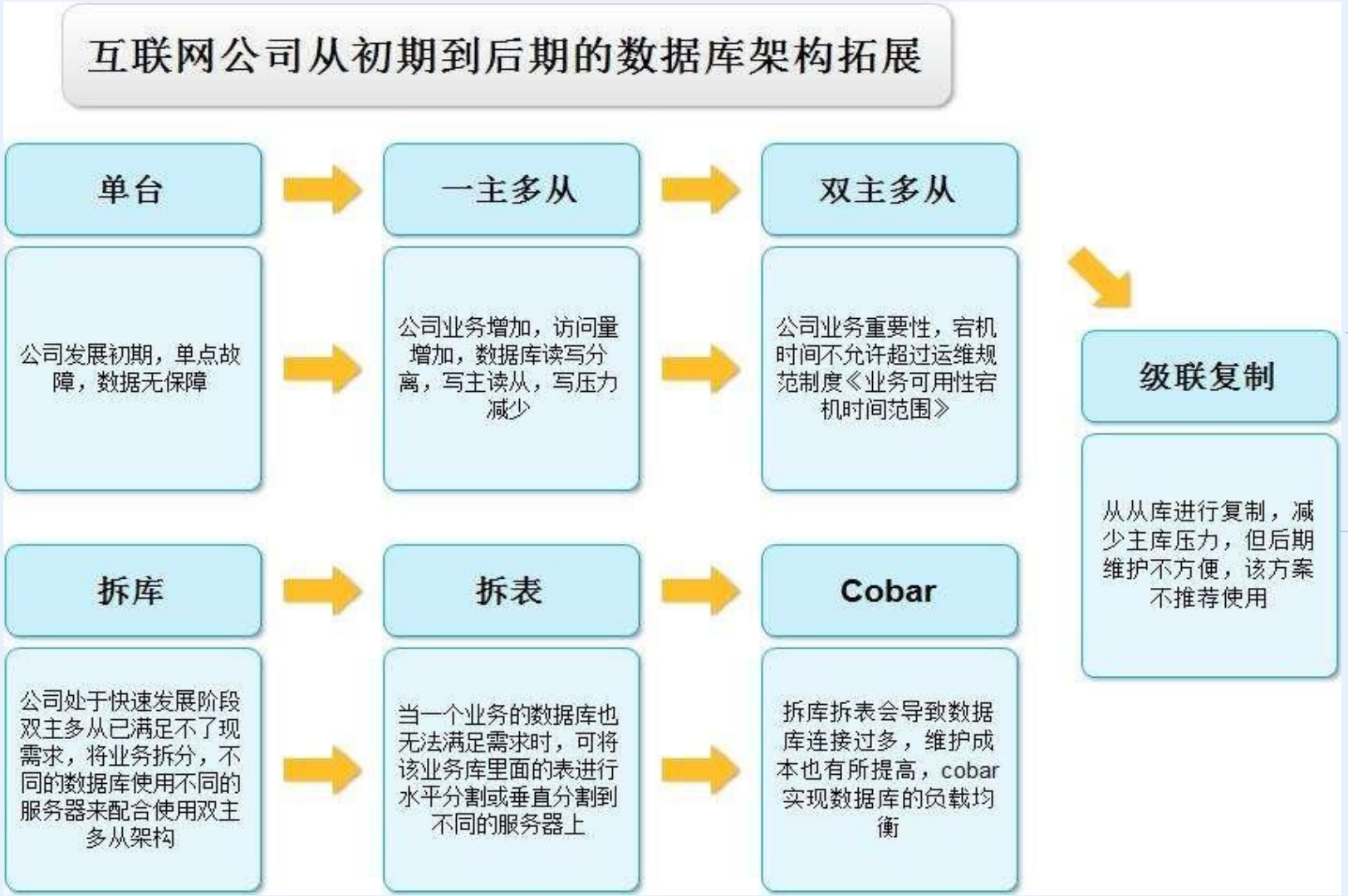
  Neo4j、OrientDB、InfiniteGraph、GraphDB...

数据模型	图结构
典型应用	• 专门用于处理具有高度相互关联关系的数据 • 适合于社交网络、模式识别、依赖分析、推荐系统以及路径寻找等问题
优点	• 灵活性高，支持面向对象思维 • 提供快速的查询和分析，支持复杂的图迭代算法 • 支持深度关联分析和高效的聚合分析
缺点	复杂性高，只能支持一定的数据规模



图形数据库

NoSQL产生的原因——应用需求的变迁



应用需求：
- 海量数据管理；
- 高并发；
- 高可扩展和高可用性

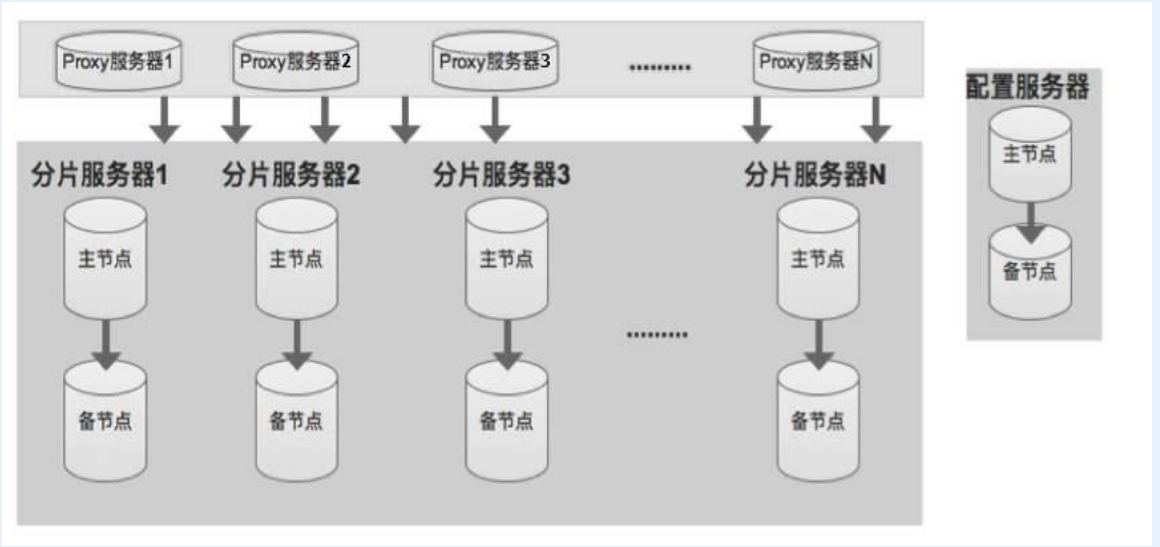
不要求：
- 不要求严格的数据库事务
- 不要求严格的读写实时性
- 不需要大量的复杂查询

传统数据库开源架构下的使用瓶颈

- **主备数据库：主从同步技术**

传统HA方案存在的问题：
- 半同步极端场景可能丢数据；高可用或高可靠，目标二选一
- 从库太多，导致复制延迟，集群性能受损

- **数据库分片技术**



分片带来的问题：
- 增加业务复杂性（部署、管理、配置），无法从根本上解决问题。
- 扩容问题 和 动态数据迁移问题。

RDBMS业界主备集群方案对比（2016年前后）

数据库集群	同步技术	功能、性能
Oracle主主集群（共享存储架构）	• Logical Standby、 • Stream、 • 11g Physical Standby(Active Data Guard)	支持一个主库最多有9个备库 • 保护最大化； • 可用最大化； • 性能最大化
MySQL主从 / 读写集群（无共享）	• 基于行的复制、基于语句的复制、混合型复制 • 半同步复制 • 基于组的复制	• 最多5台备机 • 数据复制：异步、秒级 • 不支持自动故障转移 • 可能有几分钟的数据损失
SQL Server读写集群（无共享）	• 发布订阅（表级） • 日志传送（数据库级别）、 • 事务复制（表级）、 • Always On（数据库级别）	• 数据同步：几秒 • 基本都无法实现实时数据同步。
DM7读写分离集群（无共享）	本地归档、实时归档、 同步归档、异步归档、 即时归档	• 支持一主多备，最多8台备机； • 性能提升：一主2备，性能最高可接近单机性能的3倍；一主8备，最高可达单机的7倍； • 支持秒级的故障快速切换

RDBMS业界主备集群方案对比小结

可扩展性

至多可扩展至8 - 10台备机（一般采用无共享架构），系统吞吐量接近成倍提升。

事务特性

完全保证事务ACID特性，对应用完全透明。

存储容量

每增加一台备机，数据所需存储容量翻一倍。

故障切换、同步时间

秒／分钟级完成故障切换，秒／亚秒级主备同步。

数据备份

数据多份冗余备份。

其他缺陷

备机越多，对写事务延迟影响越大。

集群中数据切片、负载均衡、主机的自动切换方案等，可能需要其他方案／应用作为补充。

2.4 NoSQL数据库 vs. RDBMS

- NoSQL数据库与关系数据库的比较

比较标准	RDBMS	NoSQL	备注
数据库原理	完全支持	部分支持	RDBMS有关系代数理论作为基础 NoSQL没有统一的理论基础
数据规模	大	超大	RDBMS很难实现横向扩展，纵向扩展的空间也比较有限，性能会随着数据规模的增大而降低 NoSQL可以很容易通过添加更多设备来支持更大规模的数据
数据库模式	固定	灵活	RDBMS需要定义数据库模式，严格遵守数据定义和相关约束条件 NoSQL不存在数据库模式，可以自由灵活定义并存储各种不同类型的数据
查询效率	快	可以实现高效的简单查询，但是不具备高度结构化查询等特性，复杂查询的性能不尽人意	RDBMS借助于索引机制可以实现快速查询（包括记录查询和范围查询） 很多NoSQL数据库没有面向复杂查询的索引，虽然NoSQL可以使用MapReduce来加速查询，但是，在复杂查询方面的性能仍然不如RDBMS

2.4 NoSQL数据库 vs. RDBMS（续）

- NoSQL数据库与关系数据库的比较（续表）

比较标准	RDBMS	NoSQL	备注
一致性	强一致性	弱一致性	RDBMS严格遵守事务ACID模型，可以保证事务强一致性 很多NoSQL数据库放松了对事务ACID四性的要求，而是遵守BASE模型，只能保证最终一致性
数据完整性	容易实现	很难实现	任何一个RDBMS都可以很容易实现数据完整性，比如通过主键或者非空约束来实现实体完整性，通过主键、外键来实现参照完整性，通过约束或者触发器来实现用户自定义完整性 但是，在NoSQL数据库却无法实现
扩展性	一般	好	RDBMS很难实现横向扩展，纵向扩展的空间也比较有限 NoSQL在设计之初就充分考虑了横向扩展的需求，可以很容易通过添加廉价设备实现扩展
可用性	好	很好	RDBMS在任何时候都以保证数据一致性为优先目标，其次才是优化系统性能，随着数据规模的增大，RDBMS为了保证严格的一致性，只能提供相对较弱的可用性 大多数NoSQL都能提供较高的可用性

2.4 NoSQL数据库 vs. RDBMS（续）

- **总结： 关系数据库和NoSQL数据库各有优缺点，彼此无法取代！**
  - 关系数据库应用场景：电信、银行等领域的关键业务系统，需要保证强事务一致性。
  - NoSQL数据库应用场景：互联网企业、传统企业的非关键业务（比如数据分析） 采用混合架构。

- **案例**：亚马逊公司使用不同类型的数据库来支撑其电子商务应用：
  - 对于“购物车”这种临时性数据，采用键值存储会更加高效；
  - 当前的产品和订单信息则适合存放在关系数据库中；
  - 大量的历史订单信息则适合保存在类似MongoDB的文档数据库中。

2.5 代表性数据库演化与发展趋势

- **内存数据库**

  提升访问性能

- **混合引擎**

  混合负载、读写双重优化， HTAP数据库 / 多模数据库

- **MPP数据库**

  ShareNothing（SN）架构，分布式、并行

- **NewSQL数据库**

  突破传统数据库技术框架，接近NoSQL数据库技术，保持SQL数据库的ACID特性

  云数据库、智能化数据库

- **新硬件技术（GPU、IO、非易失内存NVM）**

2.5 代表性数据库演化与发展趋势

2.5.1 传统数据库的演化

1）磁盘数据库→内存数据库

Oracle+Timesten，IBM solidDB...，新一代：SAP HANA ...

2）磁盘数据库+内存数据库→混合多引擎数据库

Oracle →Oracle Database in memory, DB2+SolidDB

→HTAP混合引擎

Oracle TimesTen内存数据库（1996）

- **TimesTen**: 号称最快的OLTP数据库，具有超强可用性（Ultra High Availability），弹性扩展能力（Elastic Scalability），是一种实时动态数据的高速缓存系统，包括内存数据库的连接和数据交换技术。

Timesten能比普通数据库快10倍，主要是两个原因：

（1）数据全部在内存，不需要从硬盘上取数据；

（2）应用和Timesten可以在同一台机器上直接访问，无需经过网络TCP/IP。

由于Timesten全部数据在内存中，因此需要和常规数据库配合，最终通过把数据变化写回常规数据库实现永久保存。Oracle通过Trigger方式来保持Oracle数据和Timesten数据一致，由于Trigger很消耗资源的，必须合理使用。

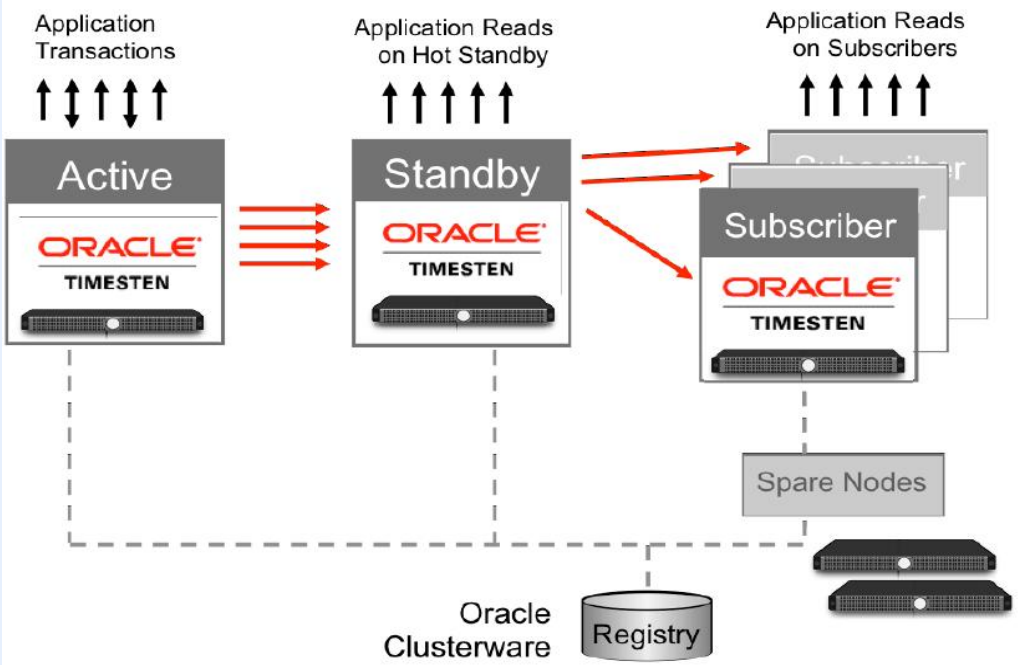
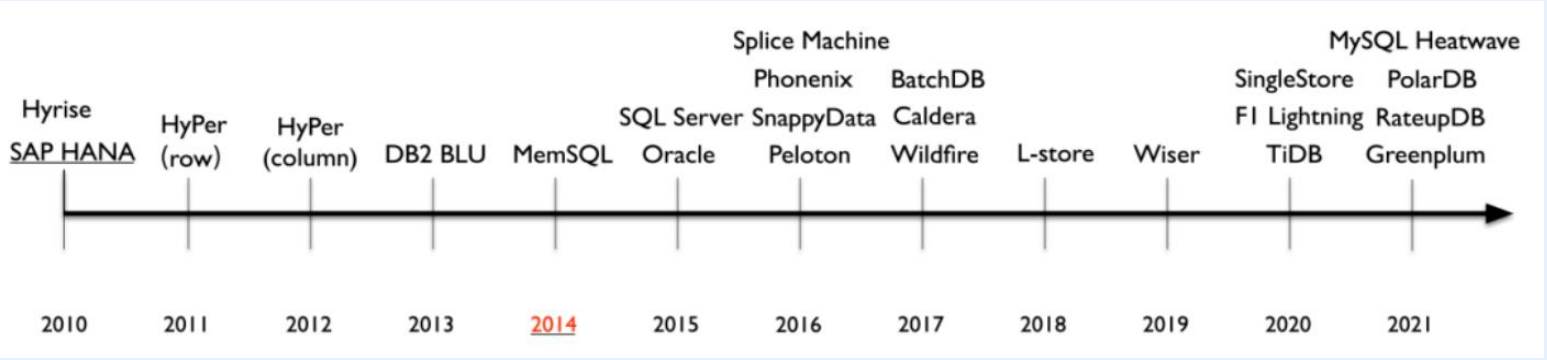
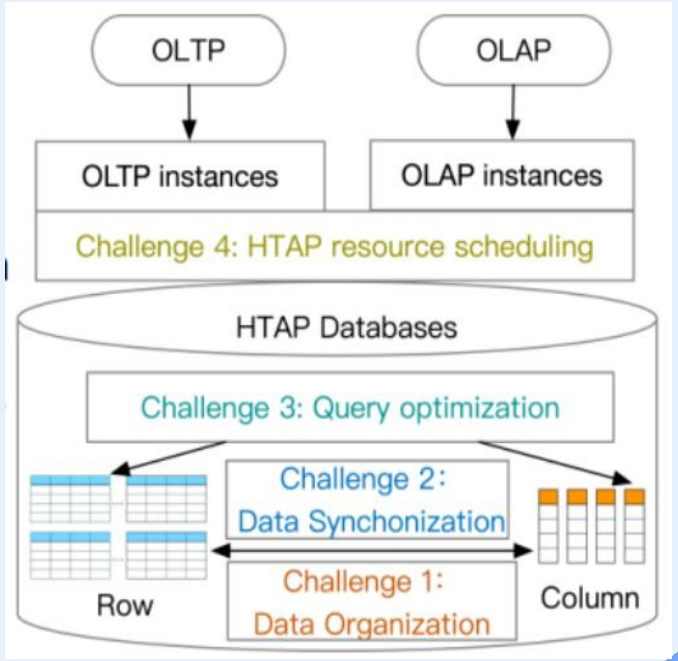


Figure 4: TimesTen A/S Pair with Clusterware

HTAP数据库的发展

- 第一阶段（2010-2014）：主要采用主列存（primary column store）的方式。如SAP HANA、HyPer、DB2和BLU等。
- 第二阶段（2014-2020）：主要扩展了以前主行存的技术，在行存上加上了列存。如SQL Server，Oracle和L-store等。
- 第三阶段（2020-present）：开启分布式的架构实现，满足高并发的请求。如SingleStore、MySQL Heatwave和Greenplum等。





Oracle Database in Memory内存列存HTAP

Oracle 数据库增添了 In-Memory 功能，能以透明的方式将分析查询速度提高若干数量级。

- 单个数据库可以高效地支持混合负载（既能以出色的性能处理OLTP事务，又能支持OLAP实时分析和报告）。——双负载能力
- 得益于“双格式”架构：支持同时以现有的 Oracle 行格式缓存（适用于 OLTP 操作）和新的纯 In-Memory 列格式（为分析处理OLAP而优化）维护数据。

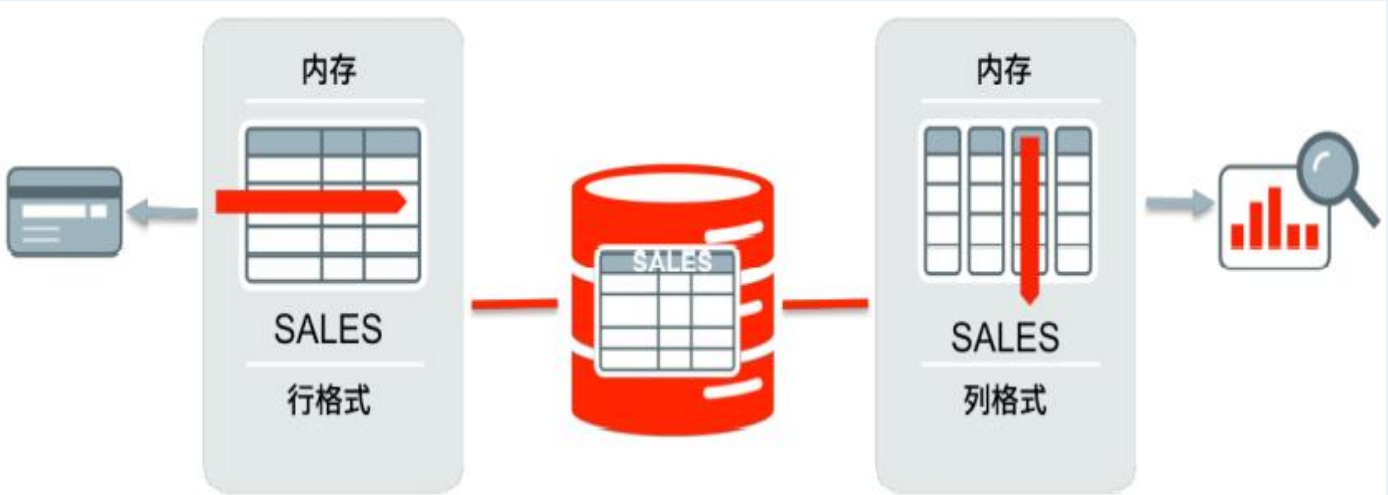
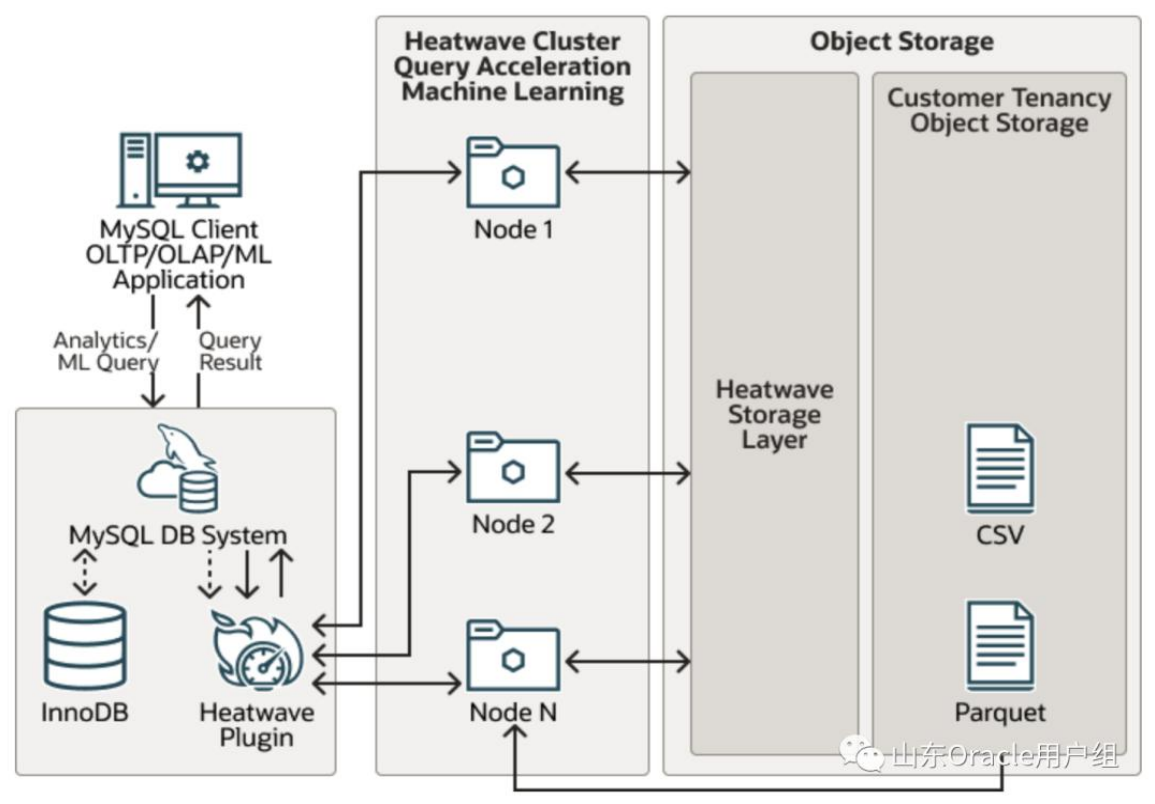


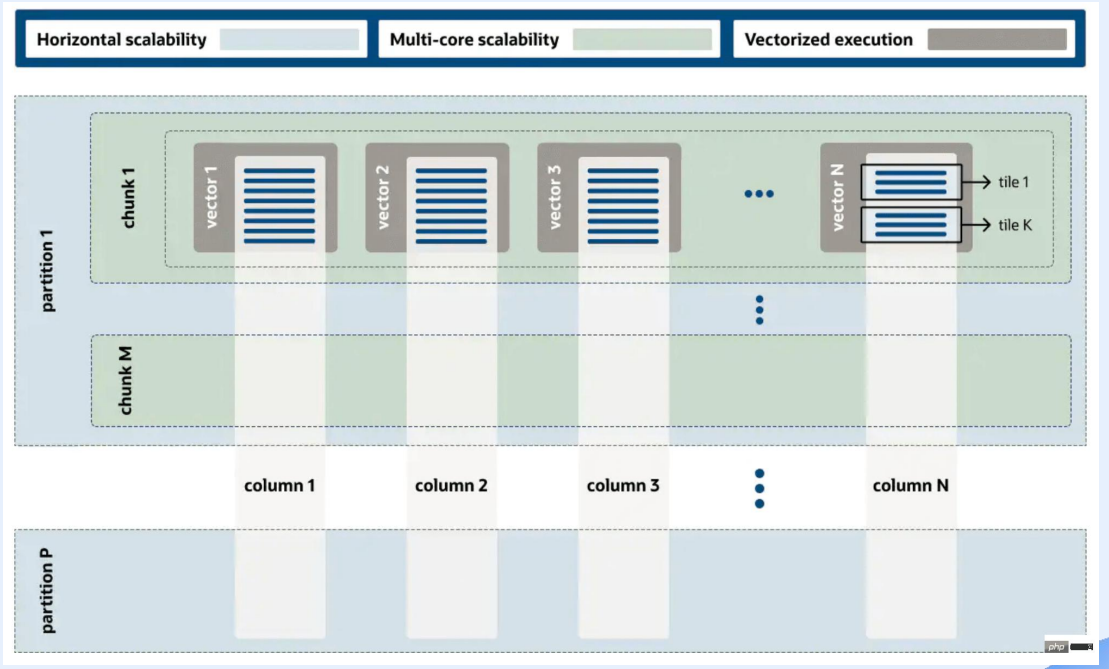
图 1.Oracle 独特的双格式架构。

In-Memory支持双格式缓存，但存储上仍保留表的单个副本，避免额外的存储成本或同步问题。

MySQL Heatwave

- HeatWave是一个分布式、可扩展、无共享、内存中、混合柱状的查询处理引擎，专为获得极致性能而设计。HeatWave架构支持OLTP、OLAP和机器学习。

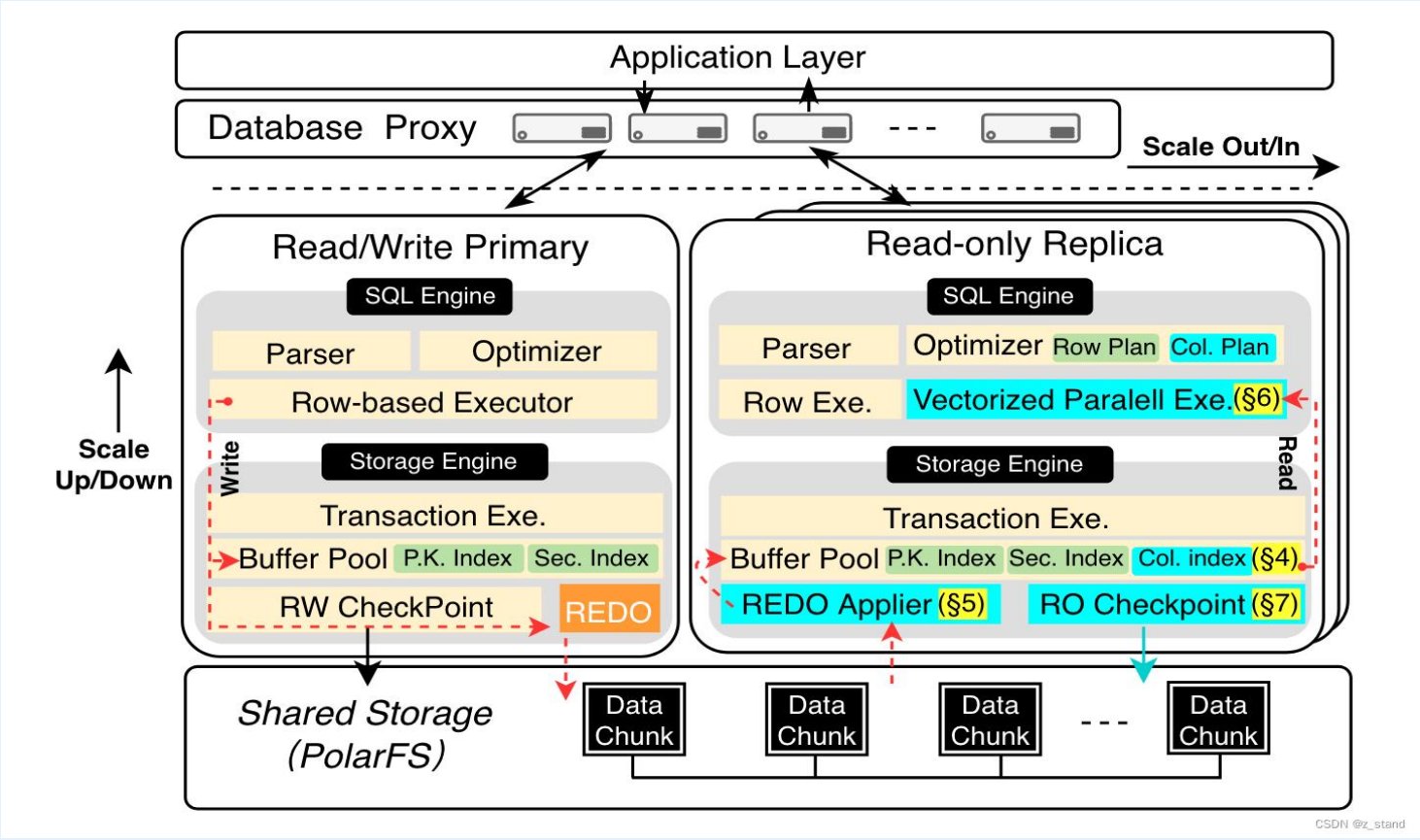




以列存方式存储在内存中，便于向量化处理

云原生HTAP数据库PolarDB IMCI

- PolarDB IMCI(In-Memory-Column-Index)

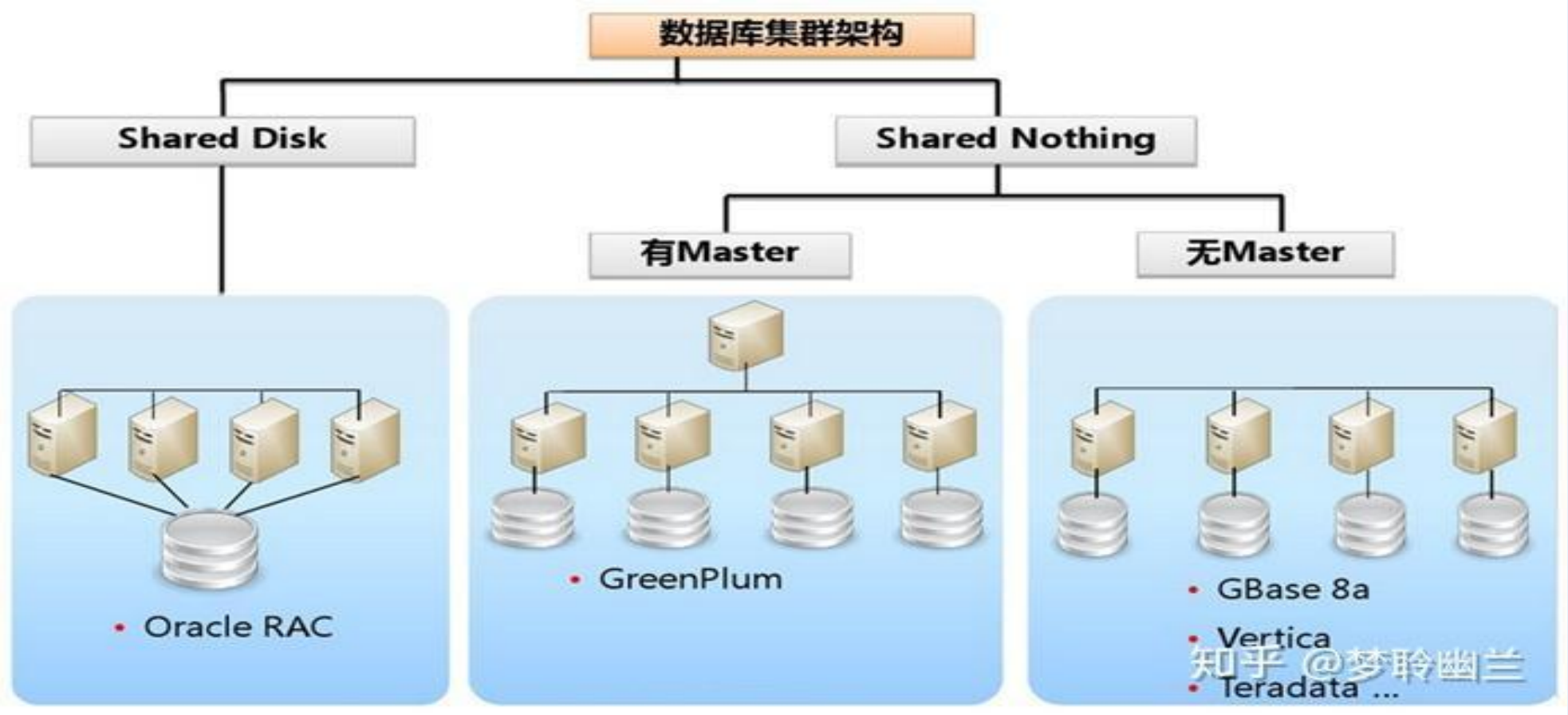


设计目标：

- 透明的SQL执行；
- 顶级HTAP；
- 对TP场景影响小；
- 数据高新鲜度；
- 优秀的资源弹性伸缩能力。

ps. 共享集群 vs. 非共享集群

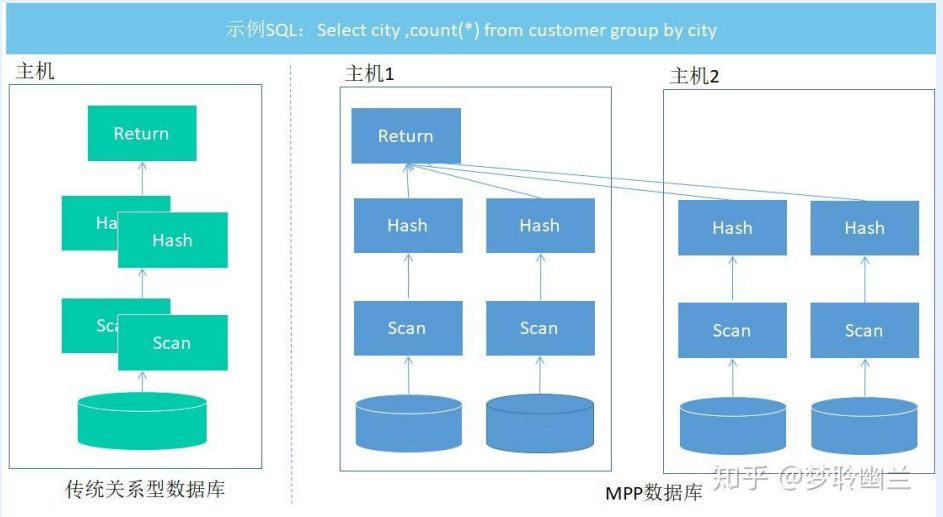
- 共享集群与非共享集群的架构差异



2.5 代表性数据库演化与发展趋势

2.5.2 MPP数据库

MPP（Massively Parallel Processing）数据库是采用无共享架构的并行分布式数据库，主要用于数据仓库类型的分析型处理负载。提供scale-out能力，通过数据分布策略和并行查询处理技术发挥集群并行处理能力。



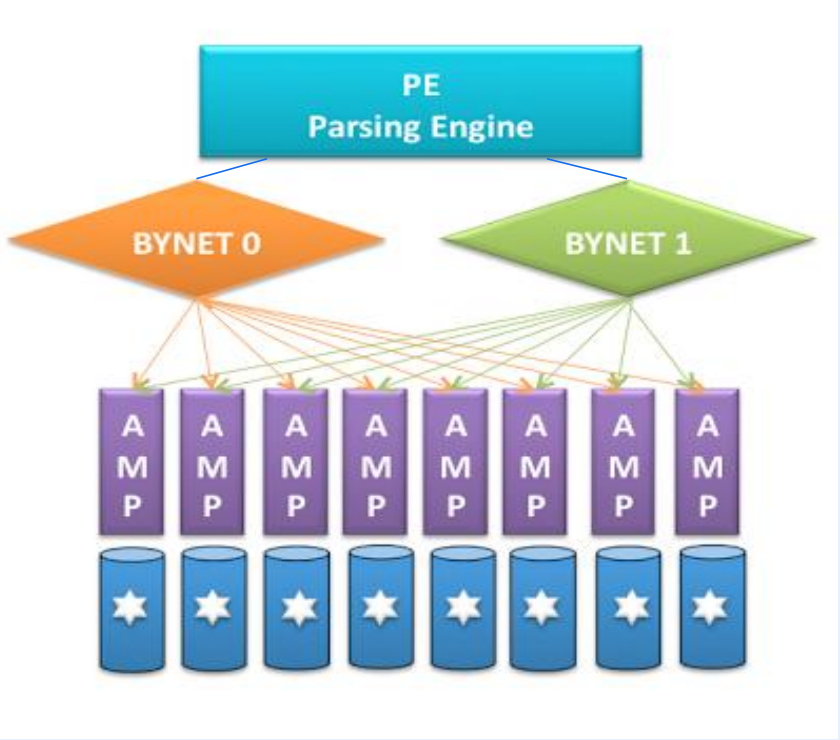
Teradata:
SN架构、混合行列分区，依据数据分区方式和查询优化技术在AMP间复制或哈希分区策略执行查询优化。
适用于大数据仓库业务，具有强大的数据处理能力和高度可扩展性、无限并行特性。

Vertica:

- 无Master的MPP，投影列存储（列之间可以有冗余）+高压缩、节点内水平分区并行处理、WOS+ROS混合负载处理。
- 更适合于实时分析场景，特别是在需要快速响应和实时数据处理的应用中表现优异。

Teradata

- Teradata采用的**SN**架构，每个节点拥有自己的硬件资源。每个AMP管理着自己的数据，协同工作，通过BYNET高速网络互联。
  - **解析引擎PE**：也称为vproc(virtual processor)，负责会话控制、SQL解析/优化、任务分发。可有**多个。**
  - **消息传输层BYNET网络**：是AMP和PE之间的桥梁（一般有两个BYNET），有负载均衡功能。
  - **访问模块处理器AMP**：管理具体数据，负责具体的磁盘的存取操作，一个AMP连接着一个/多个磁盘，是架在DISK上的桥梁。磁盘不共享，通过增加AMP来扩展系统容量。
- **混合行列分区**：将关系表水平划分为行分区，用于在SN集群内分布式存储，行分区内部再按列或列组划分为列分区。

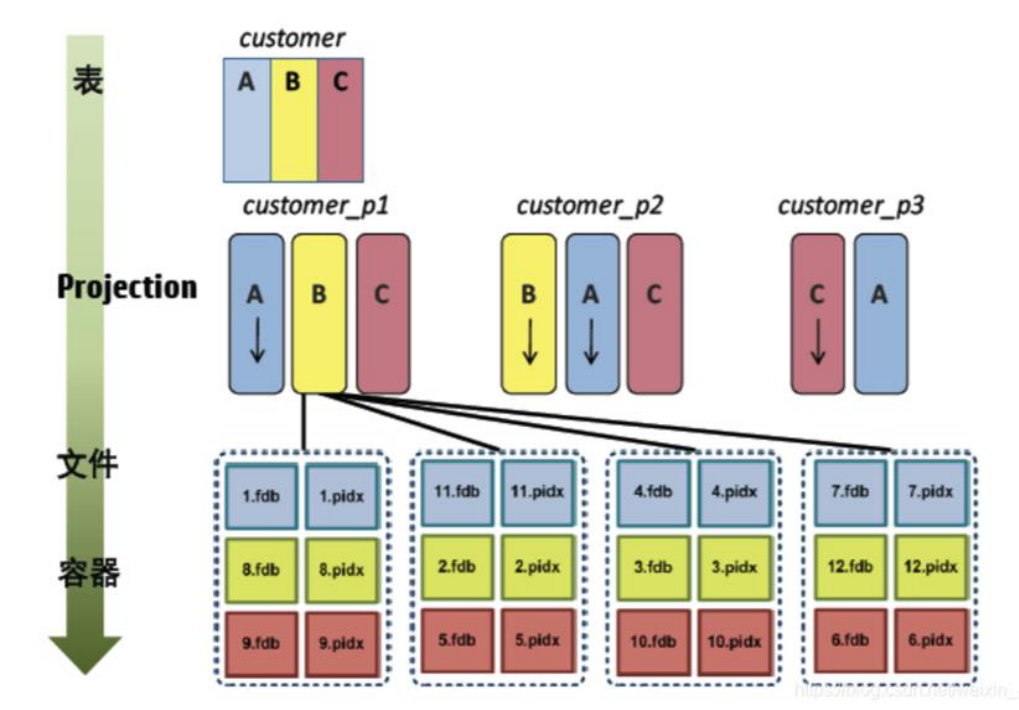


当新建一张表时，每个AMP上都会创建表的结构信息，例表名、列名、索引等信息。
理想状态下，数据库表平均的分布在所有的AMP上，以更好的利用所有节点并行处理。

参考：https://blog.csdn.net/vaychen/article/details/81216929

Vertica

- 一款基于列存储的MPP架构数据库，可以支持PB级结构化数据。
- 采用Projection列集投影存储方式：由一个或多个表中的列集组成。有序存放原始表划分为多个投影，投影之间可以有冗余，投影采用列存储。
- 节点内采用水平分区的方式将数据划分为多个存储区域，有序存放，以提高查询处理的并行性。
- 大表连接时可利用预连接投影技术。
- 列式存储，加上高压缩性能，IO能力不再是OLAP场景的瓶颈。



支持OLTP和OLAP混合负载：

- 读优存储（Read-Optimized Store，ROS），采用列存储数据压缩方式，提高分析性能；
- 写优存储（Write-Optimized Store，WOS），采用非压缩写缓存结构（行存储或列存储）。

参考：https://blog.csdn.net/weixin_43823423/article/details/87688961
https://blog.csdn.net/qq_35260875/article/details/107090268

2.5 代表性数据库演化与发展趋势

- **NewSQL数据库**

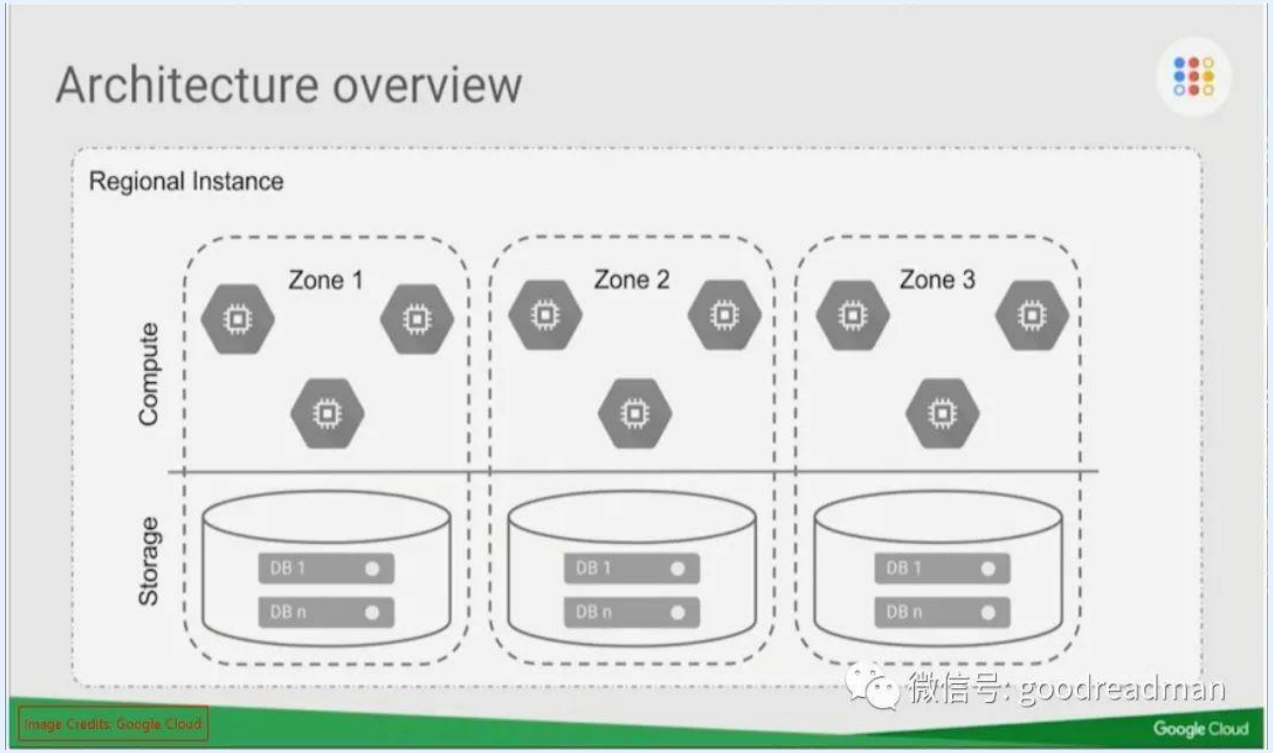
  对各种新的可扩展/高性能数据库的简称，这类数据库不仅具有NoSQL对海量数据的存储管理能力，还保持了传统数据库支持ACID和SQL等特性。

  - 无共享分布式集群：Google Spanner, VoltDB, Clustrix, NuoDB等。
  - 高度优化的SQL存储引擎：TokuDB, MemSQL等。
  - 存算分离共享存储集群：Aurora, Socrates,Tauras, AlloyDB 等。

- **基于硬件加速的新型数据库**
  - 基于GPU或者FPGA新型硬件加速：
    - MapD:基于GPU和CPU混合架构的内存数据库，数据向量化、查询并行计算技术，通过多级存储和数据压缩技术支持TB级数据处理。
  - NVM：多种持久化内存形态的总称，在SAP HANA、Oracle等大量数据库上合作。
  - IO优化：DPDK、RDMA、XDP。

全球级分布式数据库Google Spanner (2012，2017 Cloud Spanner)

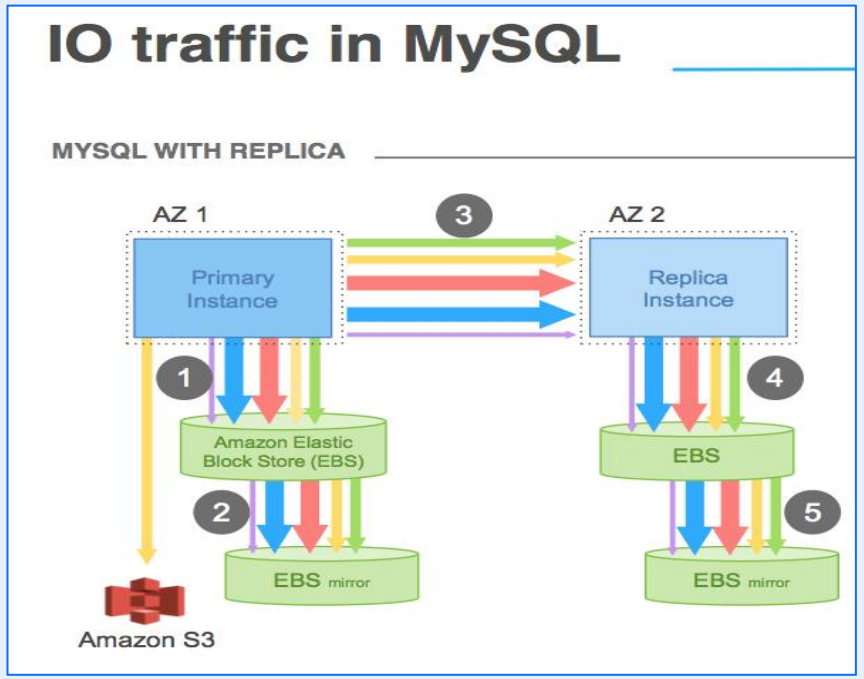
- **Spanner** 是 Google 研发的可横向扩展的、支持多版本的、可在全球范围进行分布式部署的(Globally-Distributed) 、同步进行数据复制的分布式数据库。



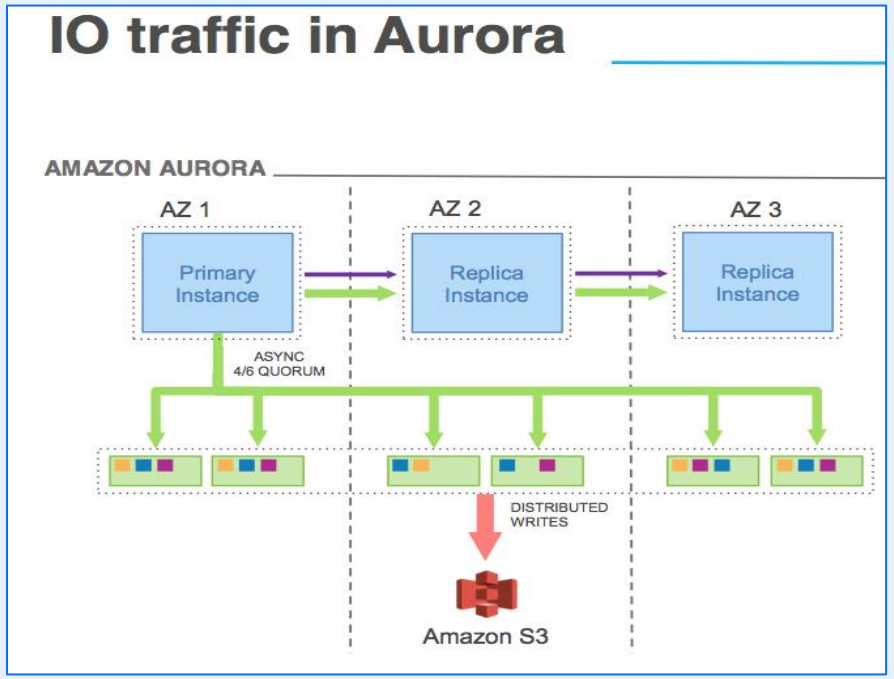
- 全球扩展 且 支持外部一致性的事务
- 计算与存储分离
- 数据散布在很多Paxos状态机中
- 存储层：数据分片、多备份（分片复制采用主从模式）
- 全球强一致性TrueTime（GPS+原子钟）
- 支持读写、只读、快照读
- 跨数据中心的数据复制
- 高可用性

Spanner对来自两个研究群体的概念进行了结合和扩充：一个是数据库研究群体，包括熟悉易用的半关系接口，事务和基于SQL的查询语言；另一个是系统研究群体，包括可扩展性，自动分区，容错，一致性复制，外部一致性和大范围分布。

云原生—共享存储型存算分离数据库Aurora



节点间传递5种类型的数据，串行

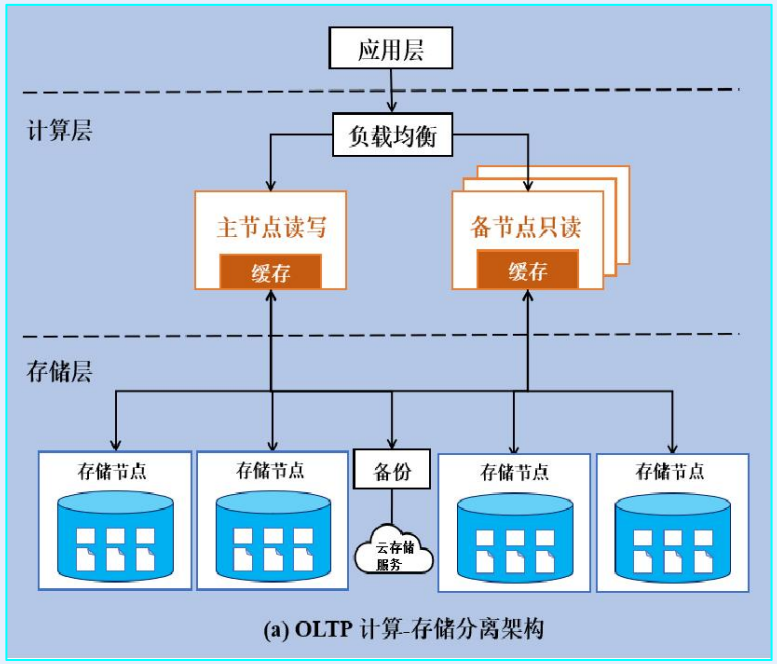


日志即数据库：基于redo日志同步；
Quorum+Gossip分布式一致性：多副本同步，日志持久化。

Aurora技术要点：存算分离，减少I/O，提高效率。

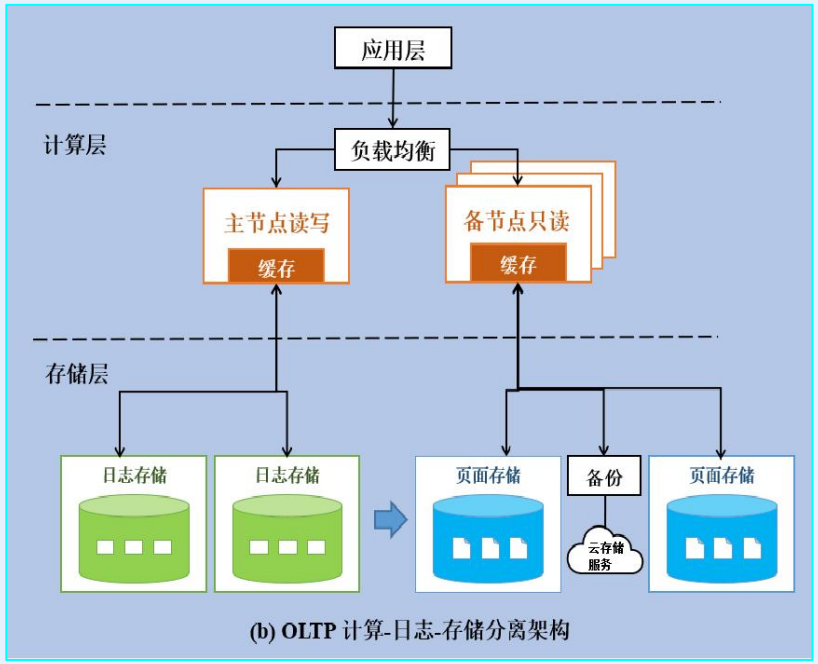
存算分离数据库分类与发展

- 存算分离数据库分类：计算-存储分离、计算-数据-日志分离、计算-缓存-存储分离。
- 开发的目标：弹性伸缩、强一致性、性能、高可用、成本



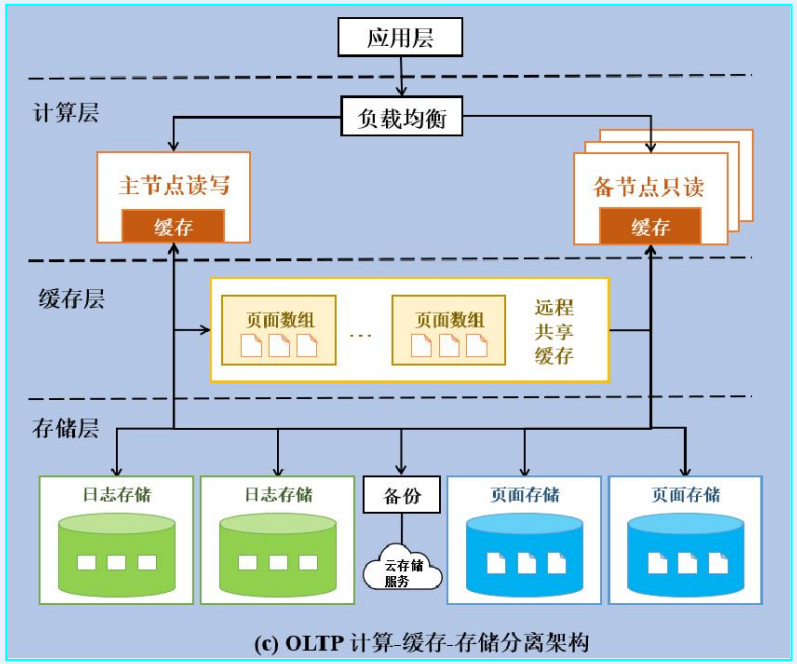
(a) OLTP 计算-存储分离架构

Amazon Aurora
PolarDB
AlloyDB



(b) OLTP 计算-日志-存储分离架构

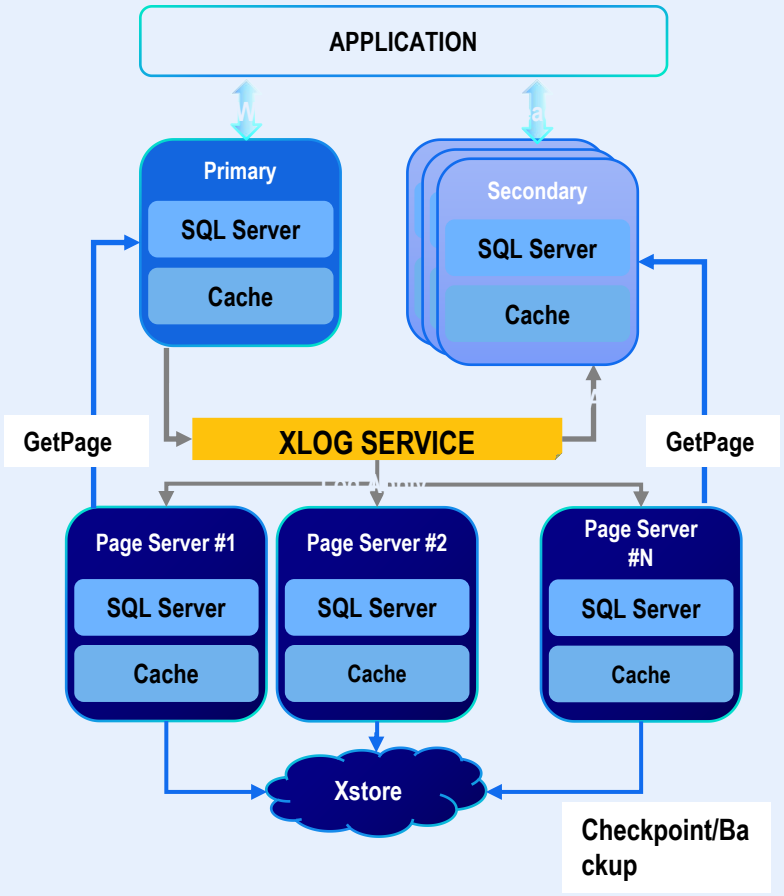
Microsoft Hyperscale
TaurusDB



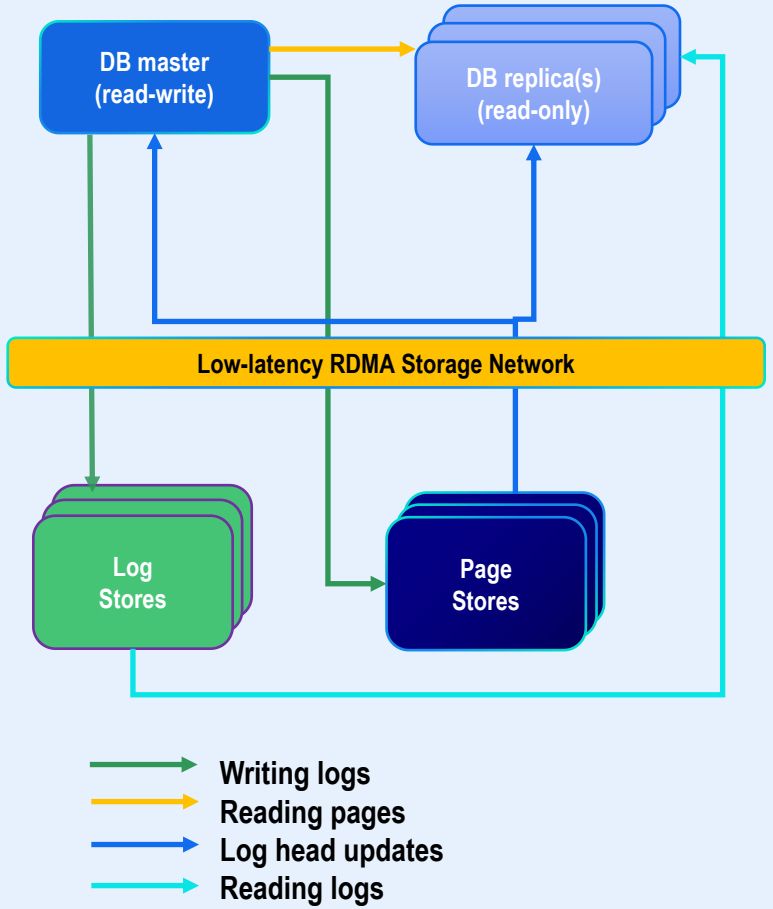
(c) OLTP 计算-缓存-存储分离架构

PolarDB Serverless
PolarDB-MP

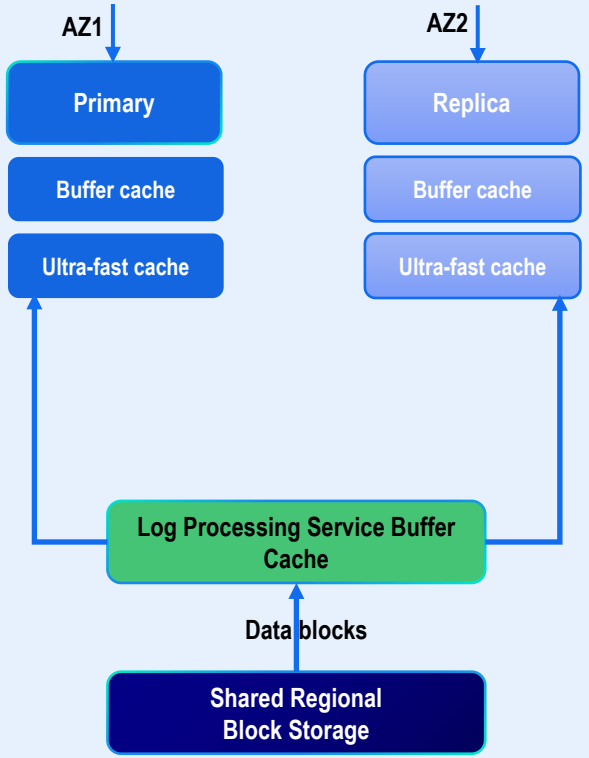
计算-数据-日志分离数据库



Socrates示意图



Tauras示意图



AlloyDB示意图

2.5 代表性数据库演化与发展趋势

- **趋势一**

  关系数据库从传统的以磁盘存储为中心的优化技术逐渐转移到以内存为主存储、分布式内存、云存储的优化设计。

- **趋势二**

  高性能内存计算推动了传统相分离的OLTP事务处理与OLAP分析处理融合到同一个数据库中，推动实时分析处理技术的发展，进一步多引擎融合。

- **趋势三**

  数据库从集中式设计转向Scale-out扩展架构，通过存算分离技术、分布式数据存储、分布式查询处理、分布式事务处理等技术支持高可扩展的数据库架构。

- **趋势四**

  非易失性内存NVM、硬件加速器、高速网络等新硬件技术推动新的查询优化和系统实现技术，支持数据库从传统硬件平台向新硬件平台的迁移。

另外：数据库智能化调优、查询优化、智能运维等研究也在蓬勃发展。

理论联系社会

关系数据模型是现代数据管理系统的重要理论和技术基础，其良好的理论基础奠定了它在数据管理领域的重要地位，也为我国社会的信息化建设提供了数十年的有力支撑。然而，时代是在不断前进发展的，科研工作也要善于寻找新常态，同时也要全面把握新形势。

互联网、电商、个人智能通讯设备的发展带来了大数据的处理需求，数据管理领域也从传统的SQL技术，发展到了NoSQL数据库，但并不意味着就应该放弃以前好的成果。不能用新的阶段背景否定以往的历史功绩，新旧阶段是相辅相成的。现有的BigTable等NoSQL数据库很多就是在设计中充分借鉴了关系数据库的设计思想，并且随着网络技术和软件理论的发展，更多的关系数据库的设计理念又进入了大数据管理系统的范畴，并由此产生了NewSQL的设计革新。

通过学习数据库管理中关系结构的应用现状，以及SQL、NoSQL和NewSQL的代表性系统及其技术特征，要学会客观认识社会需求，用实事求是的观点解决生产中的问题。

在当前大数据浪潮下，我国的有识之士和知名企业及时敏锐跟进新技术，紧密结合互联网在我国发展过程中的切实需求，取得了不菲的成绩，体现了科研工作者的奋斗精神和创新精神，也体现了面向国家服务社会的爱国情怀和责任担当。

本章小结

- 关系模型仍是数据管理最重要的数据模型。
- **回顾**

  关系数据库的基本概念、关系数据结构、关系代数
- **强化**

  SQL语言的基本语法、关系的完整性约束、强化对复杂SQL操作的理解
- **升华**
  - 面向大数据管理的SQL扩展语言知识
  - 关系数据库前沿技术
  - 掌握数据库新技术发展动向

    内存数据库、数据库一体机、列存储数据库、NewSQL数据库、GPU数据库
  - 了解大数据与新硬件时代数据库技术的发展趋势