非关系数据库知识点整理

1.关系型数据库（RDB）、NoSQL、NewSQL的区别和联系

区别：

1.关系型数据库：

优势：关系型数据库具有统一性和易用性，能够**保持数据的一致性、保证最小冗余、能够实现复杂的查询、拥有成熟的技术**。

不足：关系型数据库不适合**在分布式环境中向外扩展、难以支持高并发的读写，面对大量数据时，其性能会随着数据库的增大而急剧下降，在进行字段不固定的应用以及简单查询需要快速返回结果的情况下难以胜任。**

2.NoSQL：

优势：以放宽ACID原则为代价，采取最终一致性原则，**有利于数据的分散，能够很好地提升性能、增大规模，模式灵活而且扩展性好。**

不足：这意味着数据可能不会立即得到更新，无法达到高一致性，符合BASE模型。

3.NewSQL：吸取了RDB和NoSQL的优点，希望将ACID和可扩展性、高性能结合，降低了数据的可用性，实现了数据的一致性和分区容错性。

联系：

他们都可以大量、快速地处理并存储数据，能够满足数据库的基本要求。

2.CAP理论

C是一致性，指在分布式系统完成某写操作后任何读操作，都应该获取到该写操作写入的那个最新的值。相当于要求分布式系统中的各节点时时刻刻保持数据的一致性，系统一直有响应。

A是可用性，指一直可以正常的做读写操作。简单而言就是容户端一直可以正常访问并得到系统的正常响应。用户角度来看就是不会出现系统操作失败或者访问超时等问题一致

性：在分布式系统完成某写操作后任何读操作，都应该获取到该写操作写入的那个最新的值。相当于要求分布式系统中的各节点时时刻刻保持数据的一致性系统一直有响应

P是分区容错性，指的分布式系统中的某个节点或者网络出现了故障的时候，整个系统仍然能对外提供满足一致性和可用性的服务，也就是说部分故障不影响整体使用多节点，部分节点有故障也设关系。

CAP理论说明：**一个分布式系统不可能满足一致性，可用性和分区容错性这三个需求，最多只能同时满足两个**

3.数据库模型与CAP的对应

数据库模型分为BASE模型和ACID模型，其中nosql满足BASE模型，实现了可用性和分区容错性（AP）；RDB满足ACID模型，实现了一致性和可用性（CA），newsql满足ACID模型，实现了一致性和分区容错性（CP）

4.数据强一致性和弱一致性：

假设有三个Process分别为A，B，C，其中ABC相互独立，而且都可以实现对存储系统的read和write操作。

强一致性又叫即时一致性，假如A先写入了一个值到存储系统，存储系统保证后续A，B，C的读取操作都将返回最新值。单副本数据容易保证强一致性，多副本数据需要使用分布式事务协议。

弱一致性指的是假如A先写入了一个值到存储系统，存储系统不能保证后续A，B，C的读取操作能读取到最新值。其中**不一致性窗口**指的是从A写入到后续操作A，B，C读取到最新值这一段时间

最终一致性是弱一致性的一种特例，指的是假如A首先write了一个值到存储系统，存储系统保证如果在A，B，C后续读取之前没有其它写操作更新同样的值的话，最终所有的读取操作都会读取到A写入的最新值。
此种情况下，如果没有失败发生的话，“不一致性窗口”的大小依赖于以下的几个因素：交互延迟，系统的负载，以及复制技术中replica的个数

因果一致性指的是如果Process A通知Process B它已经更新了数据，那么Process B的后续读取操作则读取A写入的最新值，而与A没有因果关系的C则可以最终一致性

读自写一致性：如果Process A写入了最新的值，那么Process A的后续操作都会读取到最新值。但是其它用户可能要过一会才可以看到。

会话一致性：此种一致性要求客户端和存储系统交互的整个会话阶段保证Read-your-writes consistency，Hibernate的session提供的一致性保证就属于此种一致性。

单调读一致性：此种一致性要求如果Process A已经读取了对象的某个值，那么后续操作将不会读取到更早的值。

单调写一致性：此种一致性保证系统会序列化执行一个Process中的所有写操作。

5.ACID模型和BASE模型

ACID模型：

A：原子性，指一个事务中的所有操作，要么全部完成，要么全部不完成。

C：一致性，指在事务开始和结束后，数据库的完整性没有被破坏。

I：隔离性，防止多个事务并发执行时由于交叉执行而导致的数据不一致。

D：持久性，指事务结束后，对数据的修改是永久的。

BASE模型：

Basically Available--基本可用 基本可用是指分布式系统在出现不可预知故障的时候，允许损失部分可用性，如响应时间上的损失、功能上的损失等。

Soft-state --软状态/柔性事务 软状态，和硬状态相对，是指允许系统中的数据存在中间状态，并认为该中间状态的存在不会影响系统的整体可用性，即允许系统在不同节点的数据副本之间进行数据同步的过程存在延时。

Eventual Consistency --最终一致性 最终一致性强调的是系统中所有的数据副本，在经过一段时间的同步后，最终能够达到一个一致的状态。因此，最终一致性的本质是需要系统保证最终数据能够达到一致，而不需要实时保证系统数据的强一致性。
最终一致性是一种特殊的弱一致性：系统能够保证在没有其他新的更新操作的情况下，数据最终一定能够达到一致的状态，因此所有客户端对系统的数据访问都能够获取到最新的值。
在没有发生故障的前提下，数据达到一致状态的时间延迟，取决于网络延迟，系统负载和数据复制方案设计等因素

ACID	BASE
强一致性	弱一致性
隔离性	可用性优先
采用悲观、保守方法	采用乐观方法
难以变化	适应变化、更简单、更快

两者的比较

6.实现数据一致性的技术：NWR和两阶段提交协议

NWR模型：

N: 复制的节点数量，即副本数 。

W: 成功写操作的最小节点数。

R: 成功读操作的最小节点数 。

只需W + R > N，就可以保证强一致性，因为读取数据的节点和被同步写入的节点是有重叠的，同时读到的数据都是有版本号的。

如果W=N，R=1，则有较高的写延迟，而读操作几乎无延迟，具有高的一致性。（每次写都写到每个节点）

如果W=1，R=N，则有较高的读延迟，而写操作几乎没有影响，则有较高的可用性。

N越大，系统容错性越强。

两阶段提交协议：

两阶段提交协议保证了分布式事务中，要么所有参与的进程都提交事务成功，要么都取消事务，这样做可以在分布式环境中保持ACID中A(原子性)。

包含两种角色：

参与者，就是实际处理事务的机器。

协调者，就是其中一台单独的处理分布式事务的机器。

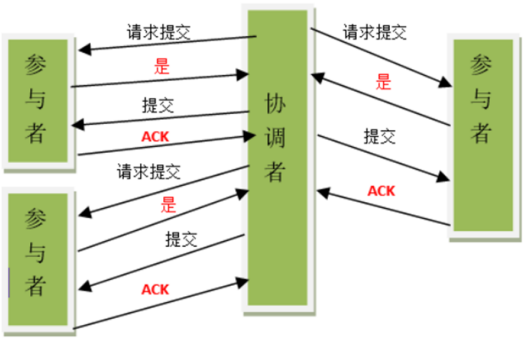
分为两个阶段：

投票阶段：

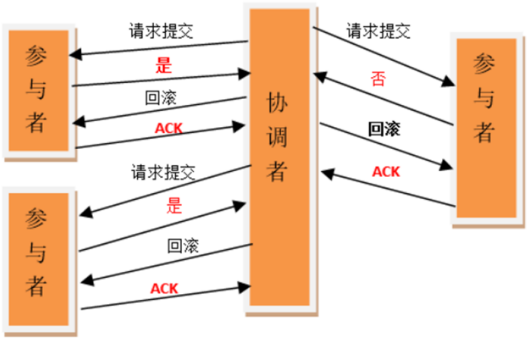
在请求阶段，协调者将通知事务参与者准备提交或取消事务，然后进入表决过程在表决过程中，参与者将告知协调者自己的决策：同意（事务参与者本地作业执行成功）或取消

（本地作业执行故障）

提交阶段，协调者将基于第一个阶段的投票结果进行决策：提交或取消。当且仅当所有的参与者同意提交事务协调者才通知所有的参与者提交事务，否则协调者将通知所有的参与者取消事务



成功的例子



失败的例子

7.非关系数据库的分类、例子与特点

存储类型	代表解决方案	特点
列存储	Hbase, Cassandra, Hypertable	按列存储，适用于数据压缩，对一个或几个字段进行查询的效率很高。
文档存储	MongoDB, CouchDB, riak	保证海量数据存储的同时，具有良好的查询性能。用类JSON格式进行存储
key-value 存储	Dynamo, Redis, Tokyo Cabinet, MemcacheDB	具有极高的并发读写性能。通过key迅速查找到value，但只能通过key查询
图数据库	Neo4j, HyperGraphDB	图形关系的最佳存储模式
对象数据库	db4o, Versant	类似面向对象语言的语法操作数据库，通过对象的方式存取数据
XML数据库	Berkerey DB XML, BaseX	高效存储XML数据，并支持XML的内部查询语法