第八章

1

- 分片策略将数据分散存储在不同节点上，以提高查询性能和并行处理能力。
- 分片策略可以根据查询和访问模式进行调整，以优化特定查询的性能。
- 复制策略通过在多个节点上复制数据片段来提高可用性和容错性。
- 复制策略确保即使某个节点发生故障，系统仍然可以通过其他节点上的副本提供数据服务。

2

1. 水平分片（Horizontal Sharding）：水平分片将数据按照某个字段（通常是分片键）的值范围进行划分，将每个范围内的数据存储到不同的节点上。例如，可以按照用户ID的范围将数据划分到不同的节点上，每个节点只负责处理自己所存储的数据。
2. 导出分片（Vertical Sharding）：导出分片将关系数据库的表按照列的划分进行分片。每个分片只包含表的一部分列，不同分片的列可以根据应用的需求进行选择。导出分片可以提高查询性能，因为在处理某些查询时，只需要访问包含查询所需列的分片。
3. 混合分片（Hybrid Sharding）：混合分片是水平分片和导出分片的结合。根据应用的需求和数据特点，可以同时采用水平分片和导出分片策略，将数据同时按照行和列进行划分。

构建关系的分片树可以帮助管理和路由分片数据。分片树是一个层次结构，其中每个节点代表一个分片，叶子节点包含实际的数据分片。构建分片树的一种常用方法是使用一致性哈希算法。一致性哈希算法将节点和数据映射到一个环形空间中，节点在环上的位置决定了它负责的数据范围。通过这种方式，可以快速确定数据应该存储在哪个节点上，并在查询时有效地路由请求到正确的节点。

3

$R \bowtie S = \pi_R(R \bowtie_{R.A=S.A} S)$

半连接运算在连接运算中能减少网络通讯开销的理由如下：

半连接运算仅返回符合连接条件的行，而不返回完整的连接结果。在分布式环境中，当关系R和关系S位于不同节点上时，如果采用传统的连接运算，需要将完整的关系R和关系S的数据传输到一台节点上进行连接操作，这会引起大量的网络通讯开销。

而半连接运算只需要将关系R的符合连接条件的行传输到关系S所在节点上进行连接操作，可以减少数据传输量和网络通讯开销。通过半连接运算，可以在分布式环境下将连接操作分布到多个节点上进行执行，从而有效降低了网络通讯开销，提高了连接操作的效率。

第九章

1

1. 分布式数据库两阶段提交（Two-Phase Commit，2PC）算法的目的是确保分布式系统中的多个节点在执行事务时的一致性。它通过协调各个参与节点的行为，使得所有节点要么都提交事务，要么都回滚事务。

基本过程如下：

- 准备阶段（Phase 1）：事务的协调者（Coordinator）向参与者（Participants）发送准备请求，询问它们是否可以执行事务，并将事务日志记录在持久性存储中。参与者执行事务的预备操作，并

将预备就绪状态（Prepared）通知协调者。
- 提交阶段（Phase 2）：如果所有参与者都准备就绪，则协调者向所有参与者发送提交请求。参与者在收到提交请求后，将事务进行提交，并将提交完成状态（Committed）通知协调者。协调者在收到所有参与者的提交完成通知后，将最终提交状态（Global Commit）发送给所有参与者，事务完成。

如果有任何一个参与者无法准备就绪或者在提交阶段出现问题，协调者将发送回滚请求，参与者将回滚事务并将回滚完成状态（Aborted）通知协调者，事务中止。

2. 2PC提交算法的缺点包括：

- 阻塞等待：在准备阶段和提交阶段，所有参与者都要等待其他节点的响应，如果其中一个节点发生故障或网络延迟，整个过程可能会被阻塞，导致性能下降。
- 单点故障：协调者是单点，一旦协调者发生故障，整个过程无法继续进行。
- 数据不一致：在提交阶段，如果协调者崩溃或网络故障，则参与者可能无法得知其他参与者的提交状态，导致数据不一致。

针对这些缺点，目前有一些改进的策略：

- 三阶段提交（Three-Phase Commit，3PC）：在2PC的基础上引入一个预提交阶段，使得参与者在准备阶段可以知道其他参与者的状态，并决定是否继续提交。这样可以减少阻塞等待的时间，并减少数据不一致的可能性。
- 基于多副本的一致性协议：使用多副本的数据复制策略，将数据复制到多个节点上，并使用一致性协议来保持副本之间的一致性，如Paxos、Raft等。这样可以避免单点故障，并提高系统的可用性和性能。
- 基于异步提交的协议：允许参与者在准备阶段完成后立即提交事务，而不需要等待协调者的指令。这可以减少阻塞等待的时间，提高并发性能。然而，需要更加复杂的协议来处理故障恢复和数据一致性的问题。

2

1. 三阶段提交（Three-Phase Commit，3PC）算法是在两阶段提交（2PC）算法的基础上引入的一种改进版本，旨在解决2PC算法的一些缺点。其基本步骤如下：

- 准备阶段（CanCommit）：协调者向参与者发送准备请求，并询问它们是否可以执行事务。参与者在收到准备请求后，首先执行事务的预备操作，并将预备就绪状态（Prepared）通知协调者。如果参与者无法准备就绪，它将发送无法准备就绪状态（CannotPrepare）给协调者。
- 预提交阶段（PreCommit）：协调者在收到所有参与者的预备就绪状态后，向参与者发送预提交请求。参与者在收到预提交请求后，会执行事务的提交操作，并将提交完成状态（Committed）或回滚完成状态（Aborted）通知协调者。
- 提交阶段（DoCommit）：协调者在收到所有参与者的提交完成通知后，向参与者发送最终提交状态（Global Commit）或回滚状态（Global Abort）。参与者在收到最终状态后，将最终状态应用于事务，并释放相关资源。

2. 相对于两阶段提交（2PC）算法，三阶段提交（3PC）算法具有以下优点：

- 3PC算法降低了阻塞等待时间：在2PC算法中，所有参与者在准备阶段和提交阶段都需要等待协调者的指令，导致可能的阻塞。而在3PC算法中，引入了预提交阶段，参与者可以在准备就绪后立即执行提交操作，减少了阻塞等待的时间，提高了性能。
- 3PC算法减少了数据不一致的可能性：在2PC算法中，如果协调者在提交阶段发生故障，参与者可能会无法得知其他参与者的提交状态，导致数据不一致。而在3PC算法中，引入了预提交阶段，即使协调者发生故障，参与者可以根据自身状态决定是否提交事务，减少了数据不一致的风险。
- 3PC算法减少了阻塞问题的影响范围：在2PC算法中，如果协调者发生故障，整个过程都会被阻塞。而在3PC算法中，即使协调者发生故障，参与者可以根据预提交阶段的结果继续执行提交或回

滚操作，降低了阻塞问题的影响范围。