可重复读简介

什么是锁机制？

锁机制是数据库管理系统（DBMS）用于管理并发访问资源的手段，目的是确保数据的完整性和一致性。锁机制通过控制不同事务对同一数据的并发访问，避免数据冲突、丢失更新等问题。

为什么需要锁机制？

在数据库中，多个事务可能会同时访问或修改同一数据。如果没有适当的控制，这种并发访问可能会导致以下问题：

1. **脏读**：一个事务读取到了另一个事务尚未提交的修改数据。
2. **不可重复读**：同一个事务中的多次读取，得到的数据在不同时间发生了变化。
3. **幻读**：同一事务内的多次查询，结果集不一致，因为其他事务插入或删除了数据。
4. **丢失更新**：多个事务同时修改同一数据时，某个事务的更新可能会被另一个事务的更新覆盖。

为了避免这些并发问题，数据库系统引入了锁机制。

锁的类型

锁机制可以细分为多种类型，按锁定对象的范围和锁定操作的性质进行分类：

1. 按操作性质划分：

- **共享锁（Shared Lock, S 锁）**：允许多个事务同时读取同一数据，但不允许修改。共享锁通常用于**读操作**，多个事务可以共享读锁，但不能对数据进行写操作。
- **排他锁（Exclusive Lock, X 锁）**：确保事务独占访问某一数据。持有排他锁的事务既可以读数据，也可以修改数据。排他锁用于**写操作**，在事务持有排他锁期间，其他事务无法读取或修改该数据。

2. 按锁定对象范围划分：

- **行锁（Row Lock）**：对某一行数据进行加锁，适用于细粒度控制并发的场景。行锁允许事务对不同的行并发操作，从而提高数据库的并发性能。
- **表锁（Table Lock）**：对整张表进行加锁，适用于需要较大范围数据一致性的场景。表锁禁止其他事务同时读写这张表。
- **页锁（Page Lock）**：对一个数据库页面（包含多个行的物理存储单元）进行加锁，通常介于行锁和表锁之间。

锁机制的实现策略

1. 乐观锁（Optimistic Locking）：

- 乐观锁的思想是“乐观”地认为冲突不会发生，所以事务在不加锁的情况下执行操作，只有在提交前检查是否有并发修改。乐观锁常用于并发写入冲突较少的场景。
- 通常通过版本号或时间戳来检测是否有其他事务修改了数据。事务提交时会检查版本号，如果版本号被其他事务修改过，则当前事务回滚。

2. 悲观锁（Pessimistic Locking）：

- 悲观锁的思想是“悲观”地认为冲突是常见的，所以在读取数据时就会加锁，防止其他事务对数据进行修改。悲观锁常用于冲突频繁发生的场景。
- 在悲观锁机制下，事务在读取数据时会申请锁，锁定的数据在事务结束之前无法被其他事务修改，确保并发访问的安全性。

锁机制的工作流程

1. **申请锁**：事务在执行读或写操作时，需要首先向数据库系统申请相应的锁。如果请求的是共享锁，则允许其他事务同时持有共享锁；如果请求的是排他锁，则需等待其他锁释放。
2. **持有锁**：一旦事务获得锁，它就可以执行对应的操作（读取或修改数据）。在持有锁的过程中，其他事务无法进行与该锁冲突的操作。
3. **释放锁**：当事务提交或回滚时，锁将被释放，其他等待该锁的事务可以继续执行操作。

锁机制避免的并发问题

通过锁机制，数据库可以解决以下常见的并发问题：

- **脏读**：通过排他锁防止未提交的数据被其他事务读取。
- **不可重复读**：通过共享锁确保同一事务中的读操作结果一致。
- **幻读**：通过行锁或表锁防止在范围查询中出现插入或删除操作引发的不一致。
- **丢失更新**：通过排他锁避免多个事务对同一数据的并发修改冲突。

常见的锁协议

数据库中常用的锁协议有以下几种：

1. **两阶段锁协议（2PL, Two-Phase Locking）**：
   - 事务分为两个阶段：加锁阶段和解锁阶段。在加锁阶段，事务可以随意加锁，但一旦开始解锁，就不能再加锁。两阶段锁协议确保了事务的隔离性和一致性。
2. **意向锁（Intention Locks）**：
   - 意向锁用于表明事务打算加锁的范围，以支持更高效的锁管理。意向锁的存在允许数据库系统为表中的某些行加锁，而不需要锁定整张表。

总结

锁机制是数据库系统中用于保证并发安全的重要手段。通过对数据加锁，数据库能够控制不同事务对数据的读写操作，避免数据一致性问题。锁的种类和机制（如乐观锁、悲观锁、共享锁、排他锁等）为数据库提供了灵活的并发控制策略，适应不同场景下的需求。