根据锁的不同分类方式，下面对不同类型的锁进行讨论。

1 锁模式

Oracle数据库自动化使用最低的限制级别提供最高程度的并发。在限制较少的情况下，有更多用户可以访问活动的数据，相反，在限制较多的情况下，事务在获得锁的时候会有更多的限制。Oracle数据库在多用户数据库中使用以下两种锁模式.。

1.1 排他锁

这种模式防止相关的资源被共享，当一个事务修改行数据时，这个事务获得一个行数据的排他锁，直到这个事务结束，排他锁才被释放，其他事务才能继续修改这个行数据。

1.2 共享锁

这种模式允许相关的资源被共享，多个事务能同时访问这些数据，共享模式保持共享锁防止一个需要排他锁的事务并行访问，在相同的资源上，可以有多个事务获得共享锁。

假定一个事务执行select ... for update语句查询单一的表行，事务将获得一个排他的行锁和一个共享的表锁，这个行锁不阻止其他会话排他锁定该行以外的其他行，同时表上的共享锁防止其他会话修改表的结构。数据库在保证数据结构不发生变化的情况下，允许尽可能多事务并行操作表的数据。

2 锁类型

锁类型分为DML锁、DDL锁、系统锁、手动锁和用户自定义锁，在使用过程中最常见的是DML锁，下面分别讨论这几种类型的锁。

2.1 DML锁

DML锁是在执行DML语句的时候产生的锁，是为了保证数据的一致性和数据表结构不发生改变的一种锁类型，用于防止多个事务对相同数据的DML操作和对表的DDL操作破坏数据

或数据结构。DML语句自动获得表锁（TM）和行锁（TX）。

1）表锁（TM）：当一个表通过insert、update、delete、merge、select ... for update子句或lock table语句修改之前，事务首先获取表锁。DML操作要求保留表锁，防止DDL操作与事务冲突；
2）行锁（TX）：是Oracle限制最大的锁，是排他锁，是表行上的锁。一个事务执行insert、update、delete、merge或select ... for update语句为修改的行获得一个行锁。行锁跟随事务一致存在，直到事务提交或回滚。行锁使用队列机制主要为了防止两个事务修改相同的行，数据库一直以独占模式锁定这些行。行锁提供很好的细粒度，由此提供了最好的并发性能。

如下表示，TM根据执行不同的SQL语句和不同的情况，又可以在排他锁和共享锁的基础上细分为6种模式。
DML表锁6中模式

锁模式号	DML表锁模式	描述
1	NULL锁	空锁
2	行共享（RS）或者子共享表锁（SS）	事务保持该锁在锁定行的表上，一个行共享锁是最低限制模式的表锁，提供表最高程度的并行
3	行排它表锁（RX）或子排它表锁（SX）	通常是事务更新表或执行select...for update产生的表锁，一个SX锁允许其他事务在相同的表中并行查询、插入、更新、删除或者锁定行。因此，SX锁允许多个事务在相同的表同时获取SX和SS锁
4	共享表锁（S）	该锁是通过事务保持的，多个事务可能并行地获取S锁，保持这个锁并不能确保事务能修改这个表。只允许其他事务查询相同的表，只有获得这个锁模式的事务能更新数据
5	共享行排它锁（SRX）或共享子排它锁（SSX）	该锁是通过事务保持的，比S锁有更多限制，只允许表有一个SSX类型的锁存在，允许其他事务以SS模式锁定表，不允许其他事务更新表
6	排它表锁（X）	该锁是最大限制的表锁，禁止其他事务执行任何类型的DML语句以及在表上放置任何其他类型的锁，允许其他事务查询数据

锁模式编号对应v$lock的lmode和request字段，锁模式编号3和4是最常见的两个模式的表锁。在表上执行update语句在表上产生锁模式为3的表锁，在表上执行create index或alter index ... rebuild语句产生锁模式为4的表锁，模式为3和模式为4的表锁可以同时存在于表上，但update操作和create index、alter index ... rebuild操作是不兼容的。

注：成功获取表锁并不代表就能执行相应的操作，是否能执行相应的操作需要根据操作类型和表的锁定情况而定。

2.2 DDL锁

DDL锁是在执行DDL语句时产生，为了防止在执行DDL命令期间其他事务修改对象结构的一种锁类型。当一个DDL操作指向一个对象，DDL（data dictionary lock）锁保护模式对象的定义。在DDL操作期间，只有被操作的对象和相关联的模式对象被锁定，数据库从来不锁定整个数据字典。

Oracle数据库自动的获取一个DDL锁，用户不能明确地请求DDL锁。例如，如果一个用户创建一个存储过程，Oracle数据库自动为存储过程引用的所有模式对象获得DDL锁，这个DDL锁是为了防止存储过程在编译完成之前引用对象被修改或Drop。

DDL锁按照模式划分为排它DDL锁和共享DDL锁。

- 排它DDL锁：防止其他会话获取一个DDL或DML锁。例如，当Alter Table执行一个表添加字段的操作时，Drop table是不允许同时执行的，同样也不允许同时向表插入数据；

排它DDL锁在DDL语句执行期间是持续的，执行完成后会自动提交。在获取一个排它DDL锁期间，如果对象的DDL锁已经被其他事务获得，那么必须等待，直到旧的DDL锁被释放才能继续执行。

- 共享DDL锁：防止破坏性DDL操作，但是允许类似的数据并发DDL操作。例如，当一个create procedure语句在运行时，事务为存储过程涉及的所有表获得共享锁，但其他事务能并行使用相同的表创建存储过程，在相同的表上获得并行的共享DDL锁，但是在有事务获得共享DDL锁的情况下，其他事务不能在存储过程涉及的表上获得排它DDL锁；

共享DDL锁在DDL语句执行期间持续保持，执行完成后自动提交，在事务执行期间，事务持有表的共享DDL锁，使得模式对象的定义不会被修改。

2.3 系统锁

Oracle数据库使用各种类型的系统锁保护内部数据库和内存结构。用户无法对其进行任何的控制，这是由Oracle内部结构来管理和控制的。系统锁能够在系统运行过程中保护内存数据结构不被破坏。主要包括Latch锁、Mutex锁和内部锁。

1）Latch锁

Latch锁是一个简单的、低级别的序列机制，它协调多用户访问共享数据结构、对象和文件。Latch锁保护共享内存资源、防止因多个进程访问导致的错误。传统的一个Latch锁保护SGA中的一组对象。

不同于DML锁和DDL锁，Latch锁不允许会话排队，第一个请求Latch锁的会话排他的获得Latch锁，如果获得Latch锁的进程没有释放资源，那么其他进程在请求相同的Latch锁时就会出现等待的情况，为了获得Latch锁，进程会不断地请求，这个过程称为Latch Spinning，当请求次数到达一个限制值，进程在继续请求Latch锁之前释放CPU资源，进入Latch锁Sleeping过程，一段时间后会苏醒并继续请求，直到获得相应的Latch锁为止。

Oracle进程获得一个Latch锁的时间非常短。例如，当处理一个员工工资更新的时候，数据库可能需要获得和释放多个Latch锁。Latch锁的实现是由操作系统决定的，特别是在一个进程是否需要请求Latch锁和等待一个Latch锁需要多长时间等方面。

某个Latch锁使用的增加意味着并发的减少，例如，过多的硬解析操作会导致Library Cache Latch锁争用，导致某部分解析必须处于等待状态。处于等待的会话不断尝试Spin空操作，将导致大量CPU的无端消耗。V$latch视图包含每个Latch详细的使用统计信息，以及每个Latch锁被请求和被等待的总次数。

2）Mutex锁

一个Mutex锁类似于一个Latch锁，一个Latch锁保护的是一组内存对象，而一个Mutex锁保护单一的内存对象。Mutex锁是从Oracle 10g R2开始引入的新技术，这并不是Oracle的发明，而是系统提供的一个底层调用，Oracle只是利用它实现串行控制的功能，并替换部分Latch锁功能，Mutex锁有以下几个优势：

- Mutex锁能减少争用的可能性。因为一个Latch锁保护多个对象，当进程尝试并行访问一组中的任何一个对象时，Latch锁可能变成一个瓶颈。通过序列化访问单独的对象而不是一组对象，Mutex锁增加了可用性；
- 一个Mutex锁比一个Latch锁消耗更少的内存；
- 在共享模式下，一个Mutex锁允许并行地被多个会话获得。

3）内部锁

是高级别的机制，比Latch、Mutex和服务器各种锁都复杂，数据库使用以下三种类型的内部锁：

- 字典缓存锁：当数据库对象被修改或使用时，这个锁被非常短地持有，是为了保护字段缓存中的内容，确保新解析的SQL语句不会看到不一致的对象定义。字典缓存锁能被共享或排他访问。当解析完成，共享锁被释放，当DDL操作完成，排他锁被释放；
- 文件和日志管理锁：用于保护各种各样的文件。例如，一个内部锁保护控制文件，以至于在一个时间点只有一个进程能改变它，另一个内部锁协调使用联机Redo日志文件，数据文件被锁定，确保单个实例数据库的数据库文件不会被多个实例同时加载，因为文件和日志锁指明了文件的状态，这些锁在长时间内都必须存在；
- 表空间和Undo段锁：用于保护表空间和Undo段。例如，RAC所有实例访问一个数据库，所有实例显示的表空间的在线或脱机状态必须是一致的。每个实例都有独立的Undo表空间，Undo段被锁定以至于只有一个数据库实例能操作这个表空间。

2.4 手动锁

Oracle数据库自动执行锁管理，确保并发情况下数据的一致性，Oracle也支持手动覆盖Oracle数据库默认隔离级别和锁机制。可以执行以下的SQL语句，在事务级别覆盖默认的Oracle数据库锁。

- set transaction isolation level;
- lock table;
- select ... for update;

执行以上的SQL语句获得相应的锁，事务结束或回滚到一个保存点之后相应的锁才被释放。如果默认的Oracle数据库锁被覆盖，那么应该确保覆盖锁的操作是正确的，不会对其他会话事务带来大范围的影响。

3 锁演示

3.1 死锁和阻塞

会话1	会话2	说明
SQL> update emp set sal=1000 2 where empno=7369;	SQL> update emp set sal=2000 2 where empno=7499;	会话1和2都拥有一个行级锁；

2 where empno=7369; 1 row updated.	2 where empno=7499; 1 row updated.	会话1和2都拥有一个行级锁；
SQL> update emp set sal=2500 2 where empno=7499;	SQL> update emp set sal=1500 2 where empno=7369;	会话彼此修改对方刚修改过的值，因资源互相被对方占用，故 都出现等待；
SQL> update emp set sal=2500 2 where empno=7499; update emp set sal=2500 * ERROR at line 1: ORA-00060: deadlock detected while waiting for resource		事务1检测到死锁，回滚并返回错误；
SQL> select empno,ename,sal from emp 2 where empno in(7369,7499); EMPNO ENAME SAL ---------- ---------- ---------- 7369 SMITH 1000 7499 ALLEN 1600		只回滚上一个事务；
SQL> commit; Commit complete.		事务1提交，并结束事务；
	1 row updated.	事务2显示已更新一行；
	SQL> commit; Commit complete. SQL> select empno,ename,sal from emp 2 where empno in(7369,7499); EMPNO ENAME SAL ---------- ---------- ---------- 7369 SMITH 1500 7499 ALLEN 2000	事务2提交，并结束事务；

3.2 使用视图查看锁

参考常用SQL