1.READPAST

ReadPast会让SQL Server在查询数据时如果遇到数据行被锁定可以跳过继续搜索数据，从而避免了锁定。

2.UPDLOCK

UPDLOCK会让SQL Server在查询数据时如果找到一条满足条件的数据行就立即将此行数据加锁，从而避免了死锁。

3.NOLOCK

NOLOCK会让SQL Server在查询数据时，忽略锁的存在而读取数据，同时读取数据也不加锁。

这在设计分布式系统时尤为重要，在设计分布式系统时我们要考虑的一个方面就是通过牺牲一致性来提高系统的并发性能，首先并不是所有情况下都要求具有很高的一致性，比如有一些提供给管理人员使用的监控系统他对系统的数据一致性和实时性的要求就不是很高，同时合理利用NOLOCK还可以减少死锁的发生机率。

比如说有一个任务系统有两个并行的流程A和B，系统规定同一个人只能做A,B两个任务中的一个，那这样一个任务系统在申请任务的时候如果不使用NOLOCK的话死锁的几率就会很高，这将在下面举一个简单的示例，此示例仅用来做简单示例。

在设计基于数据库的任务系统时如果将READPAST、UPDLOCK，NOLOCK结合将会有非常好的效果，假设我们将任务系统实现为一个基于WCF的服务。

任务系统数据表定义：

tblTask

Taskid

ProcessID

CreateTime

Userid

Status（N New,D Doing,F Finished）

接口如下：

ITaskService

string RequestTask(string userid ,string processID)

string SubmitTask（string userid ,string processID ,string taskId）

如果RequestTask的实现采用如下的关键语句的话：

```
--根据@ProcessID和@UserID来选择满足条件的任务
SELECT TOP 1 [TaskID]
FROM [dbo].[tblTask] AS A WITH(READPAST,UPDLOCK)
```

```
WHERE A.[ProcessID] = @ProcessID
    AND A.[Status] = 'N'
    AND NOT EXISTS(SELECT TOP 1 1
                   FROM [dbo].[tblTask] AS B WITH(NOLOCK)
                   WHERE B.[TaskID] = A.[TaskID]
                        AND B.[ProcessID] <> A.[ProcessID]
                        AND B.[UserID] <> @UserID)
```

//更新所选出来的任务

UPDATE tblTask SET userid= @userid,Status = 'D' WHERE TaskID = @TaskID

假设有200个客户端同时通过TaskService来申请任务的话，如果我们在SQL利用了READPAST和UPDLOCK那么每个客户端都会在短时间内获取到不同的任务，不会有阻塞，如果没有使用READPAST和UPDLOCK的话那情况就不一样了。

上面使用了两个步骤来选择任务更新任务，因此需要连个数据库通信的开销，如果能利用OUTPUT的功能在一个SQL中执行选择并更新那应该是最好的方案了，具体如下：

```
DECLARE @UserID VARCHAR(10)='001'
DECLARE @ProcessID INT = 1

DECLARE @ReturnTasks TABLE(
    [TaskID] INT NOT NULL,
    [CreateDate] DATETIME
);

UPDATE [dbo].[tblTask] WITH(READPAST,UPDLOCK)
SET [UserID]= @UserID, [Status] = 'D'
OUTPUT DELETED.[TaskID],DELETED.[CreateDate]
INTO @ReturnTasks
FROM (SELECT TOP 2 A.[TaskID]
      FROM [dbo].[tblTask] AS A  WITH(READPAST,UPDLOCK)
      WHERE A.[ProcessID] = @ProcessID AND A.[Status] = 'N'
            AND NOT EXISTS(SELECT *
                           FROM [dbo].[tblTask] AS B WITH(NOLOCK)
                           WHERE B.[TaskID] = A.[TaskID]
                                 AND B.[ProcessID] <> A.[ProcessID]
                                 AND B.[UserID] <> @UserID)
      ORDER BY A.[CreateDate] DESC) AS C
WHERE [dbo].[tblTask].[TaskID] = C.[TaskID]

SELECT * FROM @ReturnTasks ORDER BY [CreateDate] DESC
```