This document specifies a couple of ways to lock a key range. Although it needs to be mentioned that FoundationDB strongly recommends that application data modeling and access patterns should try to minimize conflicts as much as possible, the following suggestion is only for those rare scenarios.

For example, consider a scenario where there is range of keys k1, k2, ... kn, and a transaction selects a key from this range arbitrarily (say kx), reads the value of kx, and then write some value back at kx. There is an additional constraint - that only one transaction could be modifying this entire range at a given time (i.e. concurrent modifications in this range, even if on different keys, are disallowed).

To achieve this, each of the above transactions could either add an explicit ReadConflcit, or a WriteConflict on the entire range of keys (k*). From correctness or performance point of view, there is no difference between both the approaches for the above scenario. However, the more natural way to do it would be to add read conflict range for the whole range and write conflicts only for the keys that are actually being modified. Read conflict set should, in some sense, be the set of things whose values need to be kept constant between transaction start time and commit time. The write conflict set is then the set of things whose values are updated as a result of the current values in the read conflict set.