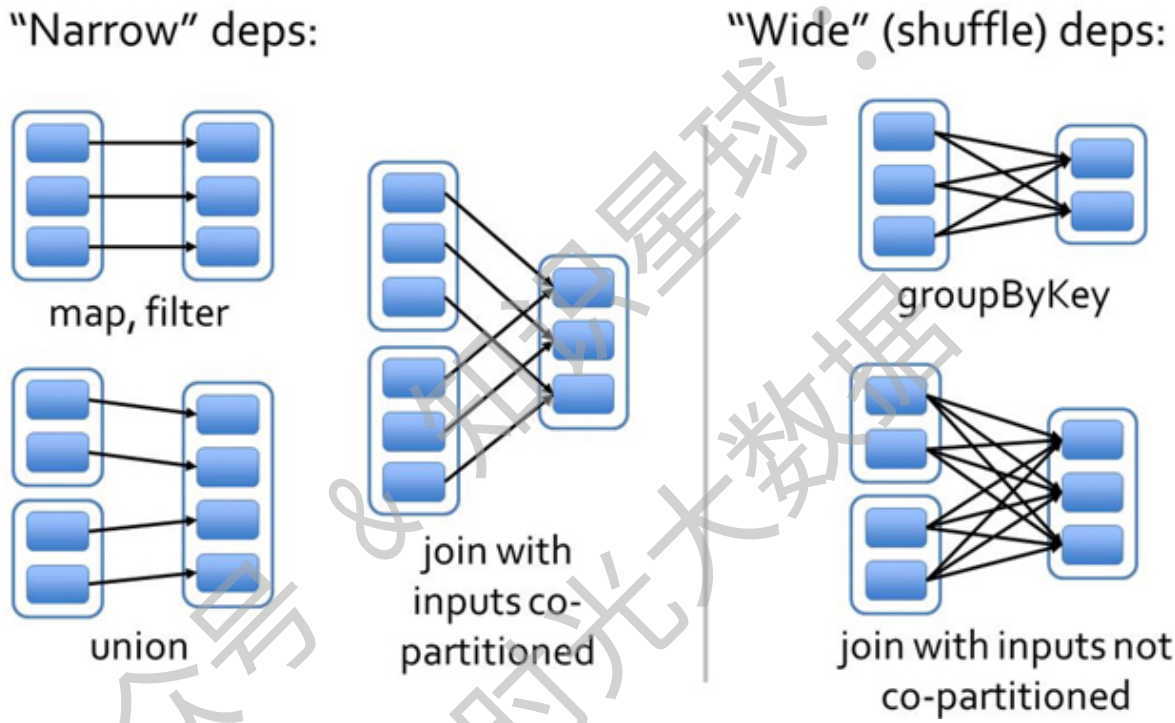
Spark join在什么情况下会变成窄依赖?

参考答案:

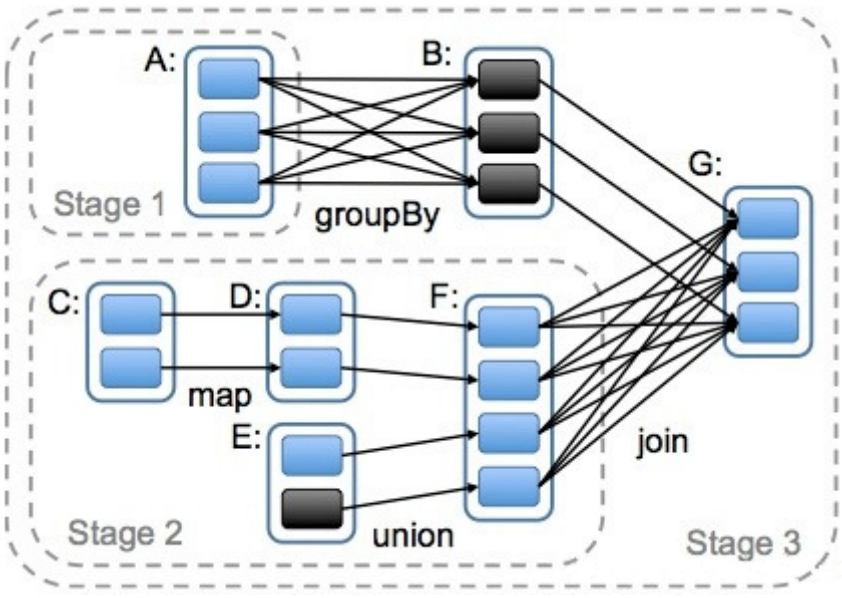
如果需要join的两个表，本身已经有分区器，且分区的数目相同，此时，相同的key在同一个分区内，就是窄依赖。反之，如果两个需要join的表中没有分区器或者分区数量不同，在join的时候需要shuffle，那么就是宽依赖。



2、划分宽窄依赖原由

需要从宽窄依赖和容错性方面考虑。

Spark基于lineage的容错性是指，如果一个RDD出错，那么可以从它的所有父RDD重新计算所得，如果一个RDD仅有一个父RDD（即窄依赖），那么这种重新计算的代价会非常小。



对于Spark基于Checkpoint（物化）的容错机制，在上图中，宽依赖得到的结果（经历过Shuffle过程）是很昂贵的，因此，Spark将此结果物化到磁盘上了，以备后面使用。

对于join操作有两种情况，如果join操作的每个partition仅仅和已知的Partition进行join，此时的join操作就是窄依赖；其他情况的join操作就是宽依赖；因为是确定的Partition数量的依赖关系，所以就是窄依赖，得出一个推论，窄依赖不仅包含一对一的窄依赖，还包含一对固定个数的窄依赖（也就是说对父RDD的依赖的Partition的数量不会随着RDD数据规模的改变而改变）。

如果不进行宽窄依赖的划分，对于一些算子随便使用，可以使用窄依赖算子时也使用宽依赖算子，就会造成资源浪费，导致效率低下；同理，当需要使用宽依赖算子时，却使用窄依赖算子，则会导致我们得不到需要的结果。

相比于宽依赖，窄依赖对优化还有以下几点优势：

- 宽依赖往往对应着shuffle操作，需要在运行过程中将同一个父RDD的分区传入到不同的子RDD分区中，中间可能涉及多个节点之间的数据传输；而窄依赖的每个父RDD的分区只会传入到一个子RDD分区中，通常可以在一个节点内完成转换。
- 当RDD分区丢失时（某个节点故障），spark会对数据进行重算。
    - 对于窄依赖，由于父RDD的一个分区只对应一个子RDD分区，这样只需要重算和子RDD分区对应的父RDD分区即可，所以这个重算对数据的利用率是100%的；
    - 对于宽依赖，重算的父RDD分区对应多个子RDD分区，这样实际上父RDD 中只有一部分的数据是被用于恢复这个丢失的子RDD分区的，另一部分对应子RDD的其它未丢失分区，这就造成了多余的计算；更一般的，宽依赖中子RDD分区通常来自多个父RDD分区，极端情况下，所有的父RDD分区都要进行重新计算。
    - 如下图所示，b1分区丢失，则需要重新计算a1,a2和a3，这就产生了冗余计算(a1,a2,a3中对应b2的数据)。