Minimal Basis

我们原来计算一组attribute的closure的方法是迭代

有没有第二种解法呢？

canonical cover: //canonical 最简洁的

G与F是两组FD

G是F的cononical cover 当：

1.G equivalent to F 就是G cover F, F cover G ,也就是F+=G+

2.G的每一个FD右手边只能有一个attribute

3.G是minimal的（最精简的），如果我们删除了G中的某些FD， 那么得到的剩下的set H 就与F不相等

从两个方面看canonical cover G 是最简洁的

1. G里的每一个FD都是required,只有这样才让G equivalent to F,少一个就不行

2. G里的每一个FD都是尽可能的小的， ABC->Y 为例

每一个左边的attribute都是必要的·

右边的attribute只能有一个

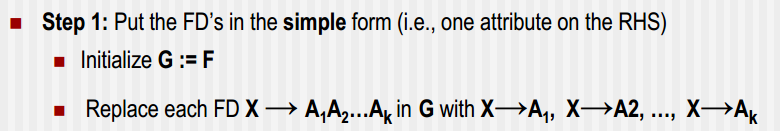
Computing canonical cover 0:12 0:15

给你一组FD, 怎么计算出他的canonical cover G

Step1:

初始化G=F

把所有FD展开成simple form



然后右边可以直接拆



step2:

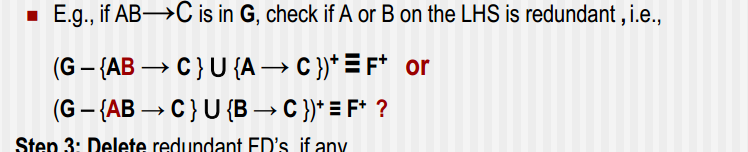
minimize每个FD的左手 ，尽量压缩

例如：G里有AB->C

看看A或B是多余的

G减去这个FD，加上一个A->C，看看能不能替代

或者加上一个B->C，看看呢能不能替代



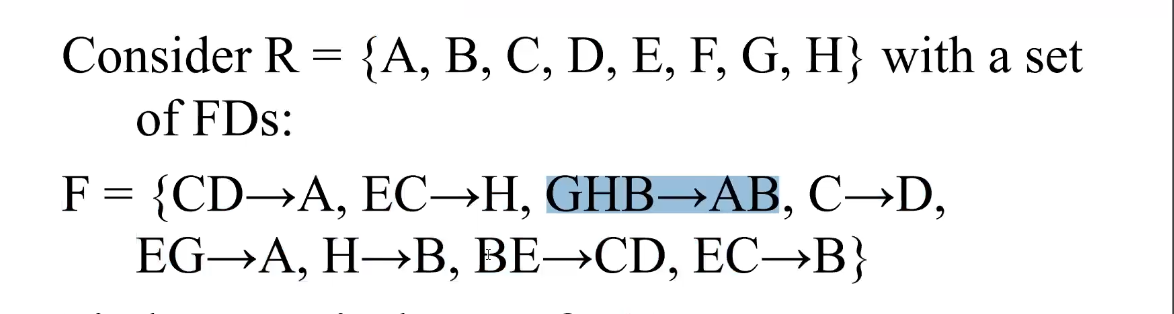
0：15

STEP3：

删去多余的FD

对于每一条X->A，看看G少了它还等不等于F

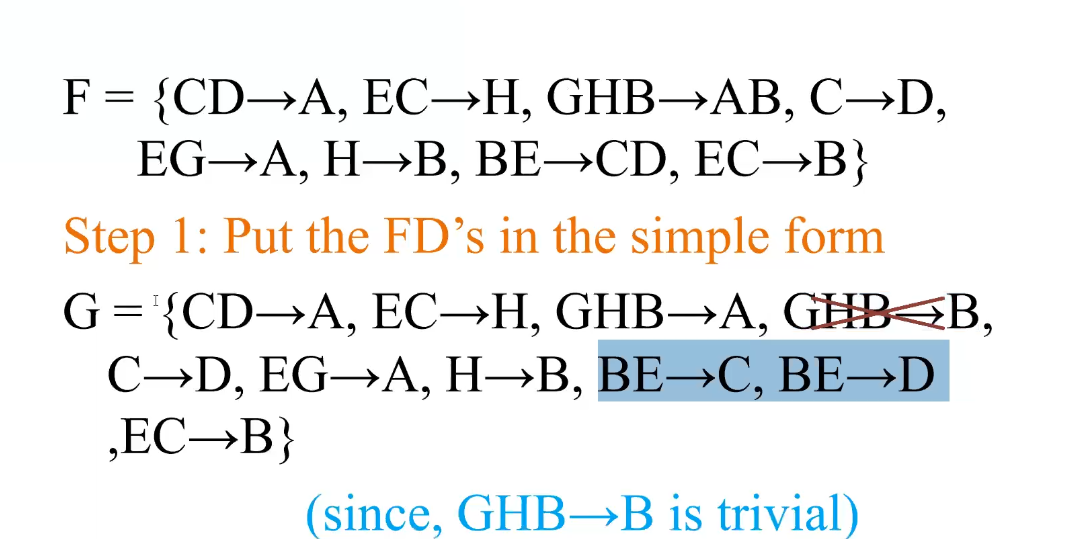




STEP1:

G=R，

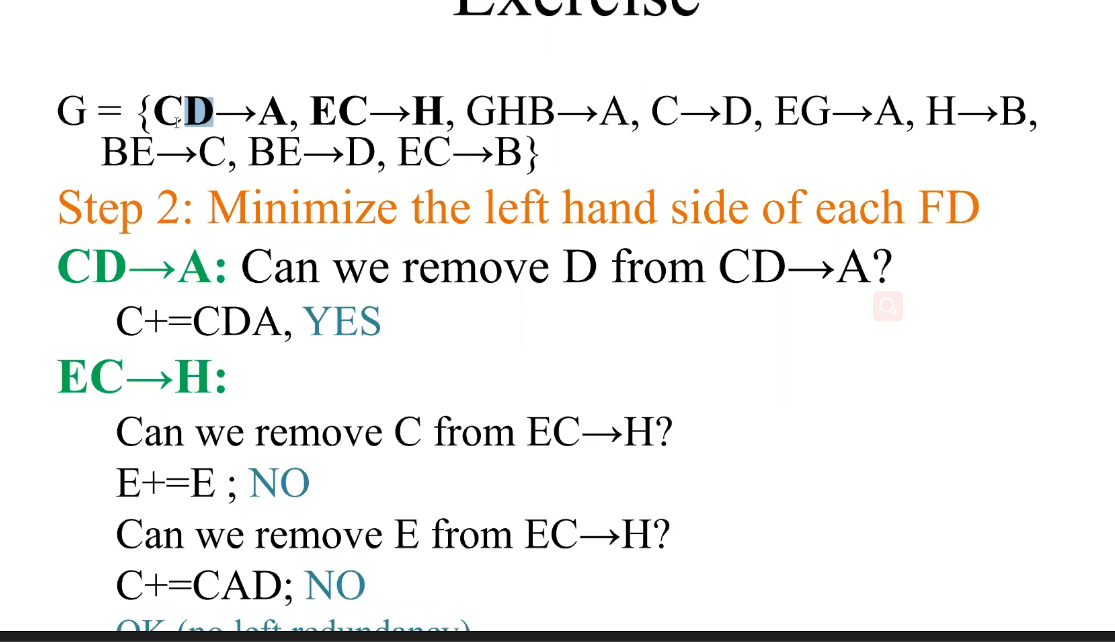
拆分，让右边变成simple form



GHB- >AB拆成GHB->A, GHB->B ,第二个可以删掉，因为trivial了

STEP2:

minimize左边，



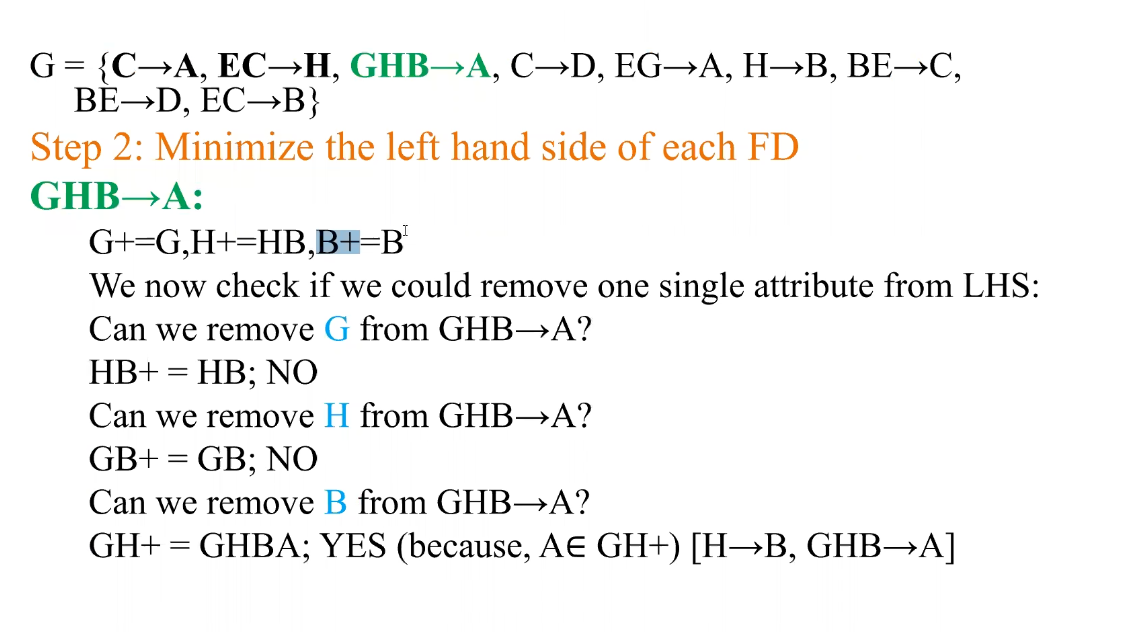
CD->A

就是看看去掉了D，还能不能推出右边的

换句话说也就是求剩下的closure

D的closure还是D，因此不能去C

去掉D,C的closure就是C->D CD->A, C+=CDA,包含了A，因此可以去掉D

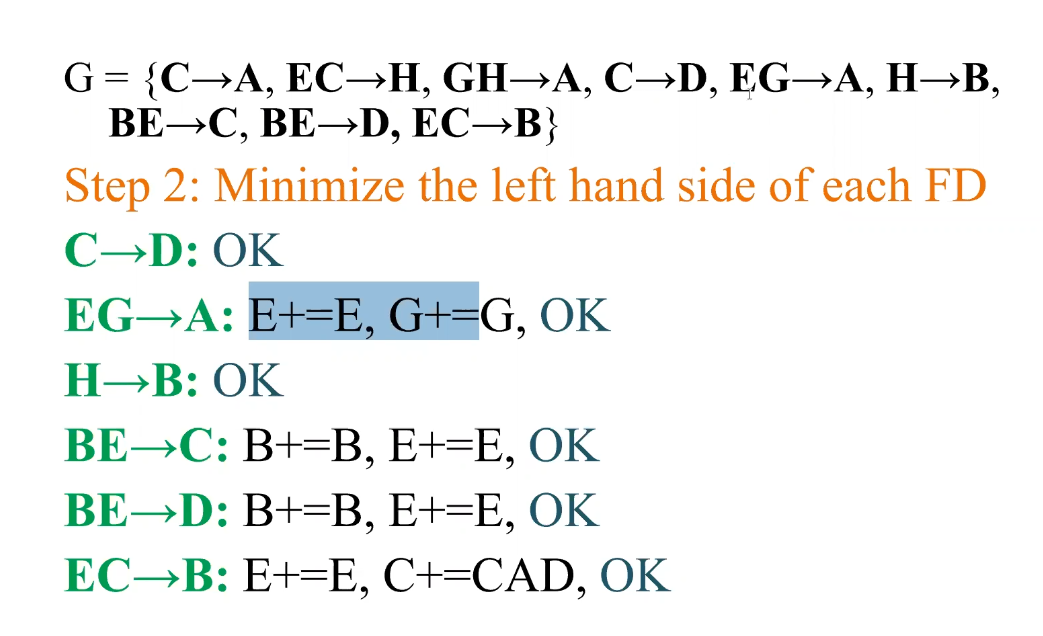


HB+=HB

GB+=GB

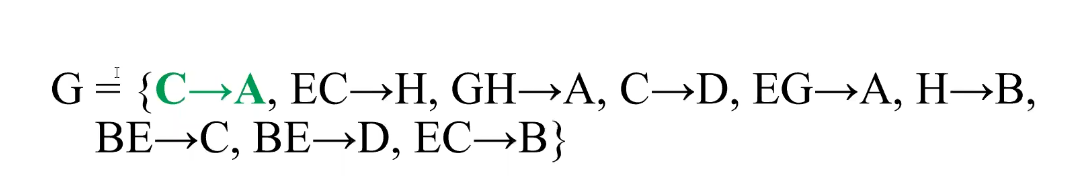
GH+=GBH=GBHA,

而A在GBHA里，舍去

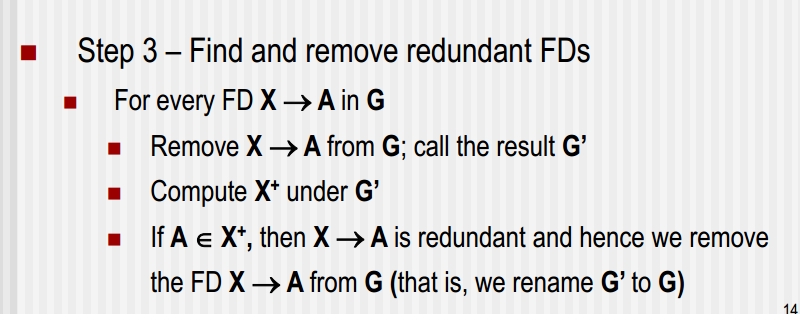


剩下的都OK

然后G变成这样



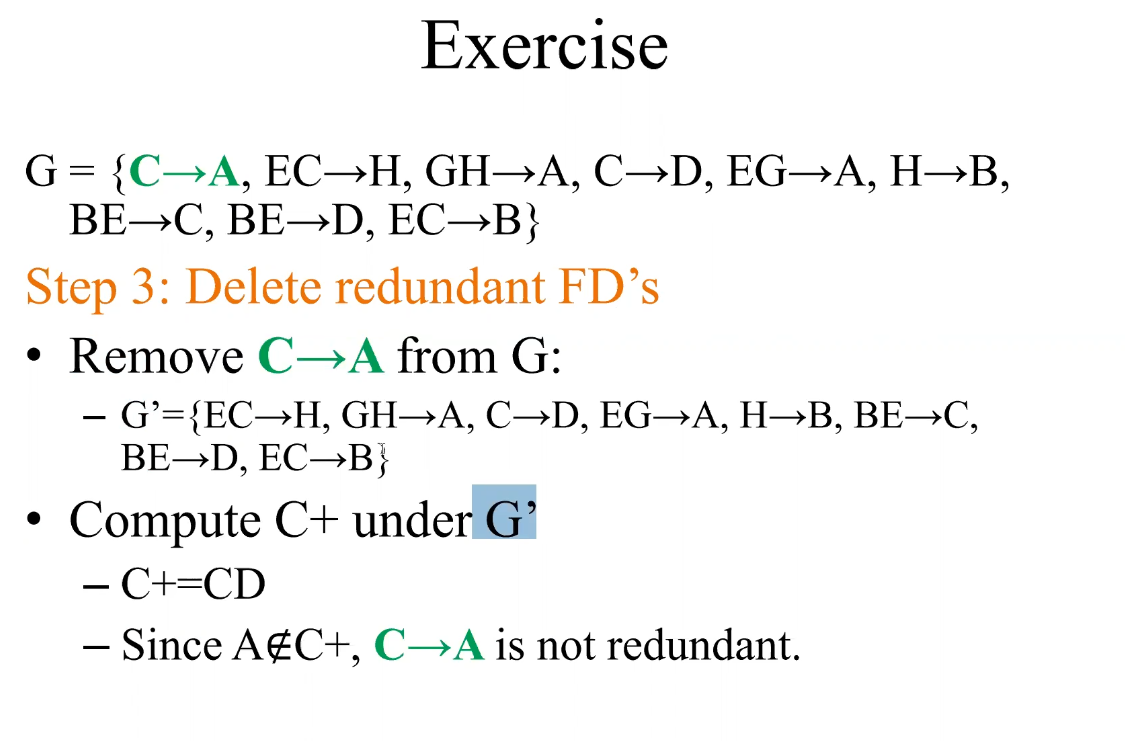
第三步减少重复的

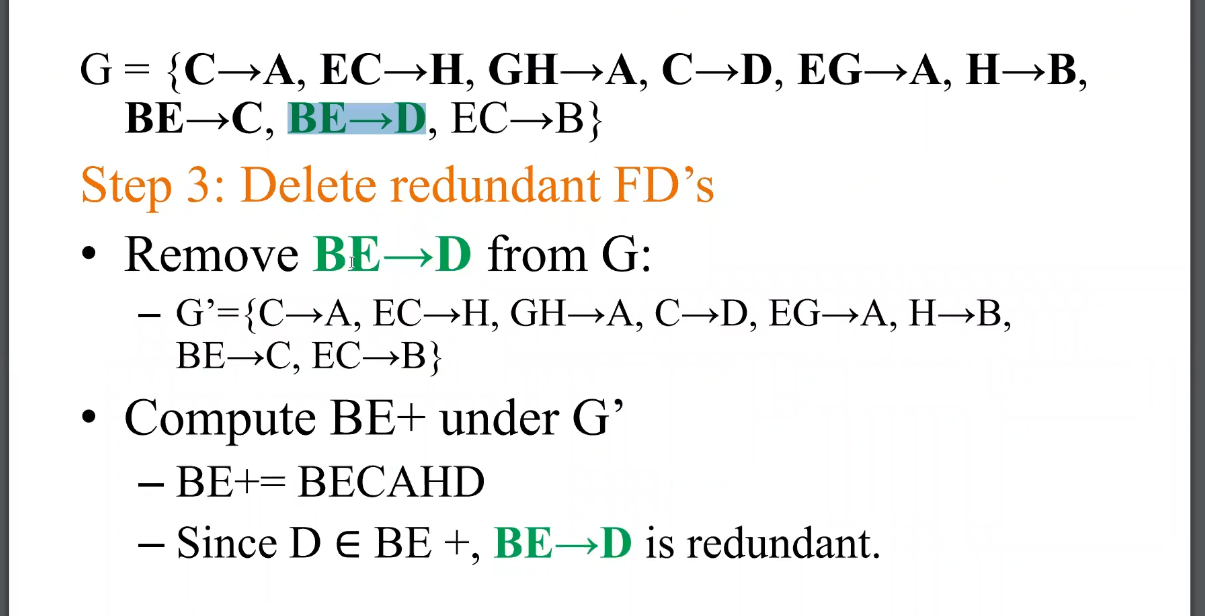


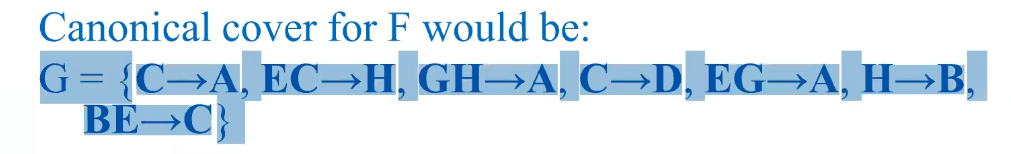
尝试着把每个FD都去掉，

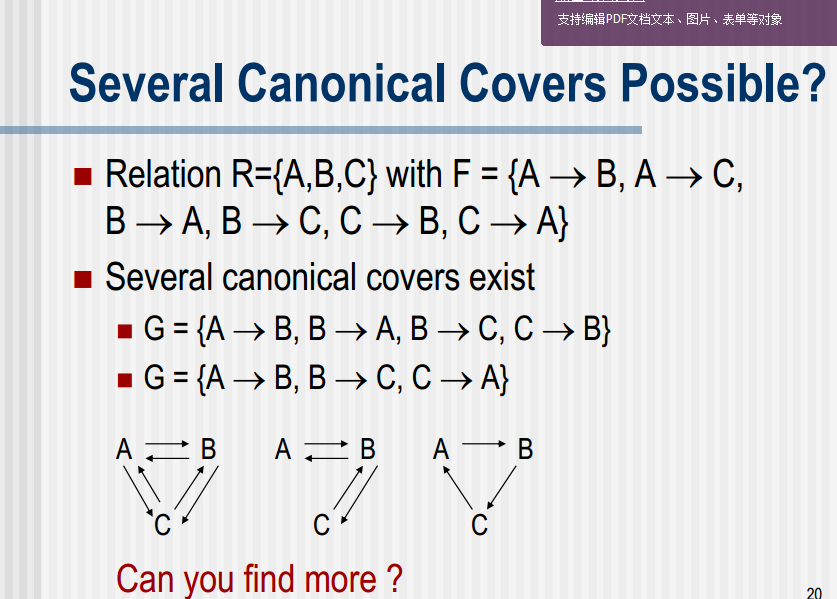
然后把去掉的FD的左边,假设他为X,求X+在剩下的G的值，

如果X+还是能推出原来的结果,A属于X+,那么我们删了它



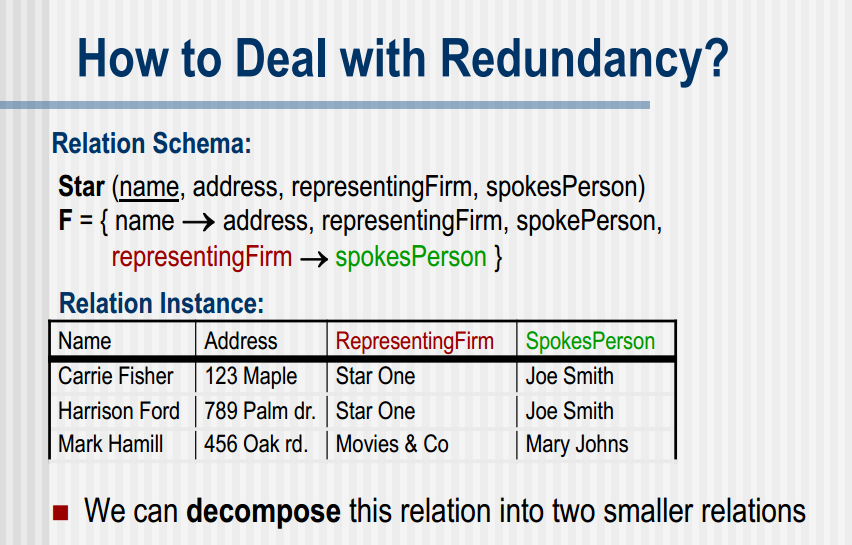






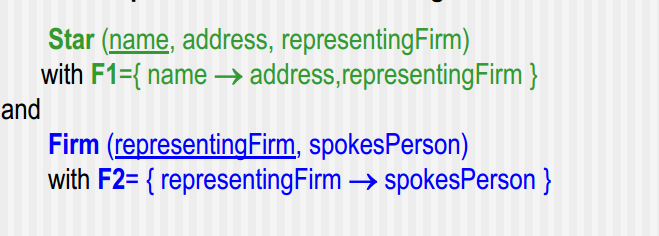
Canonical Covers是不唯一的，可以同时有多个canonical covers

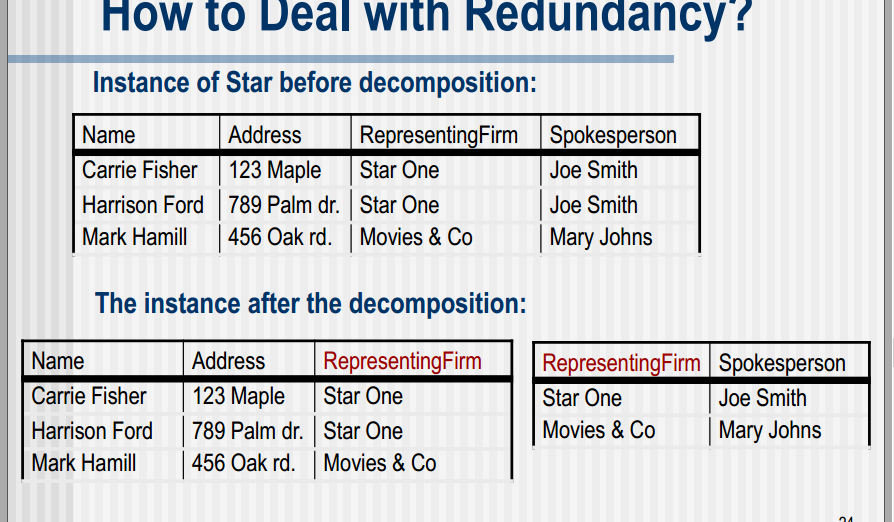
怎么解决redundancy



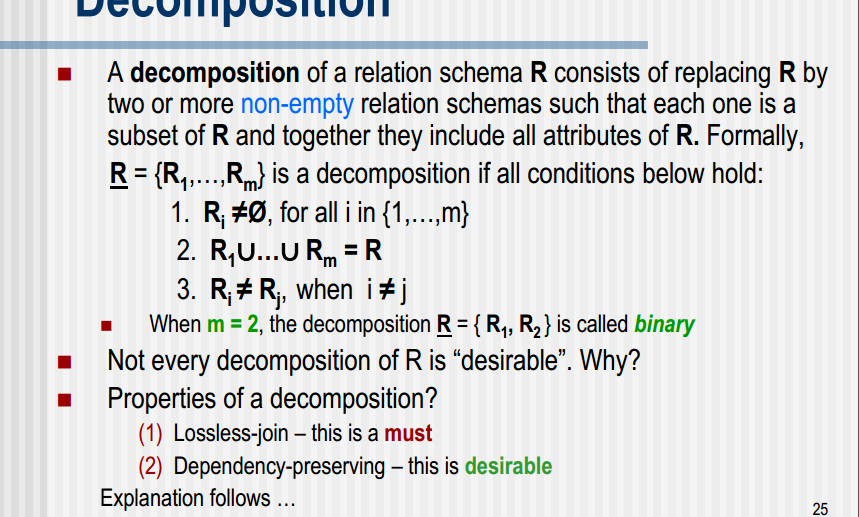
就Relation里面有了额外一组FD关系

我们可以吧这个relation拆解decompose





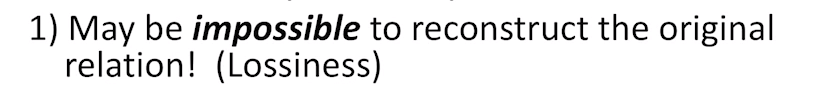
Decomposition of a relation schema R, 用两个或多个non-empty relation schemas来代替R，每一个都是R的子集且他们合起来能包含R的所有attribute

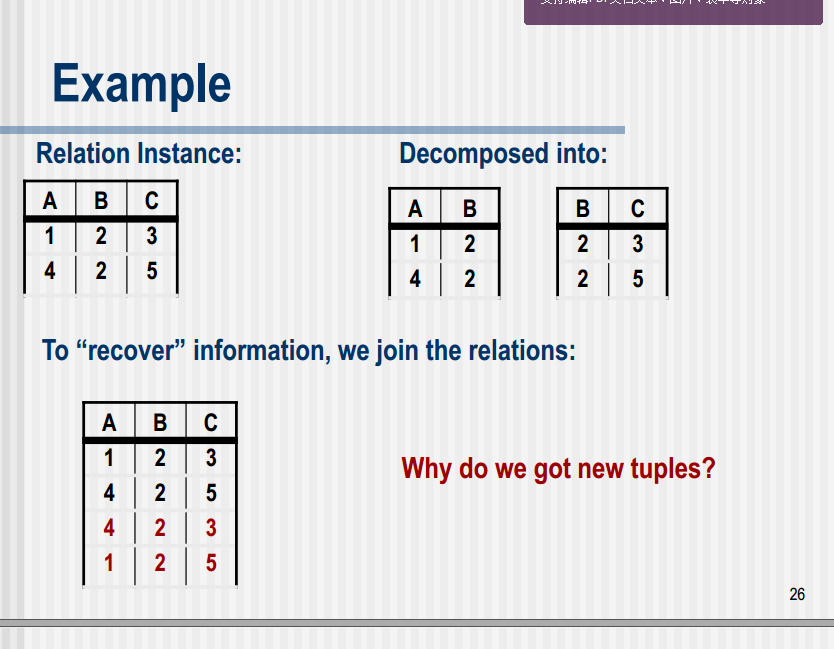


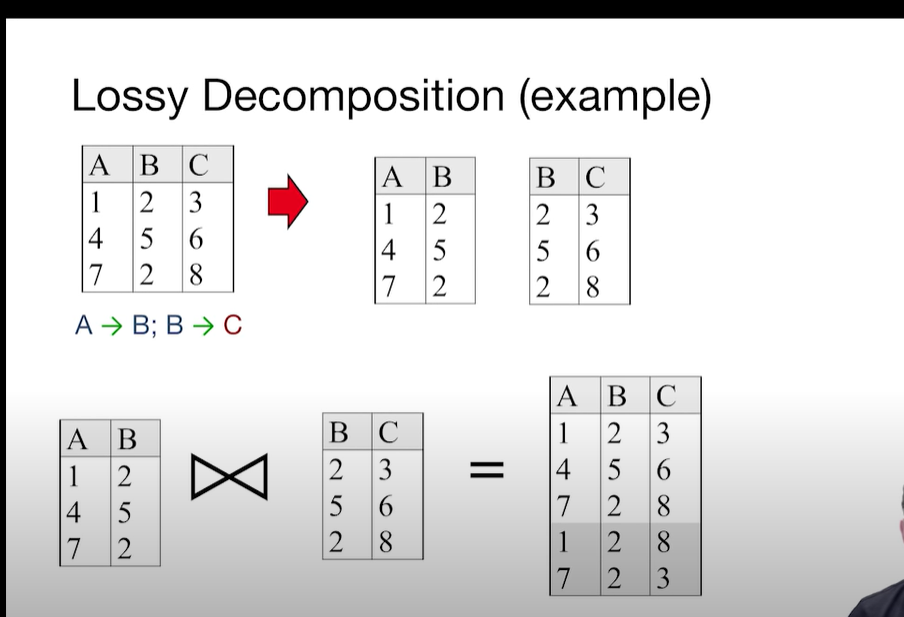
如果之拆成两个，叫做binary

’

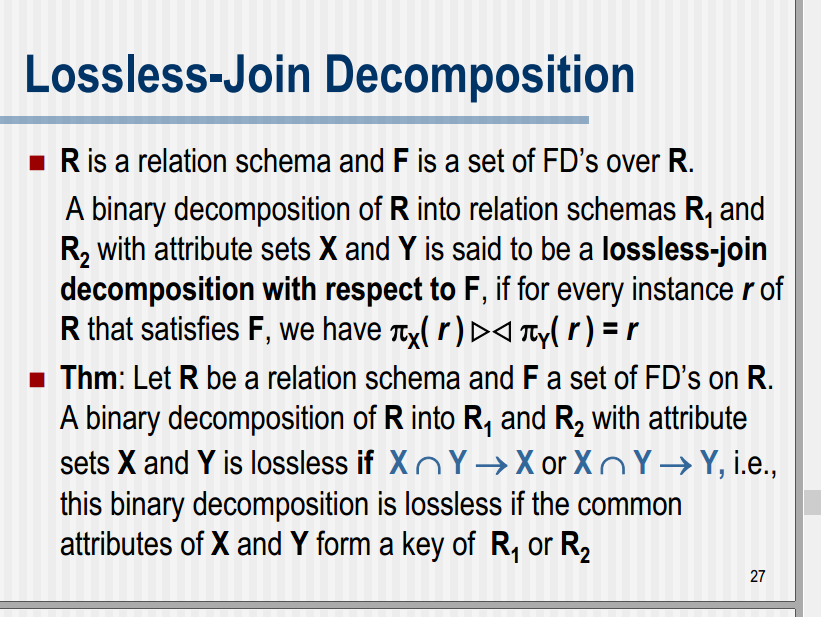
但是有时候Decompositions会导致lossiness





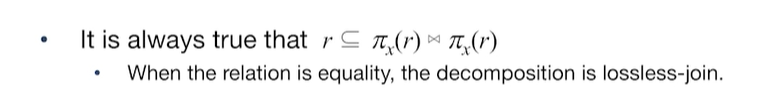


lossiness不是丢失信息，而是生成了我们不想要的结果，没法复原





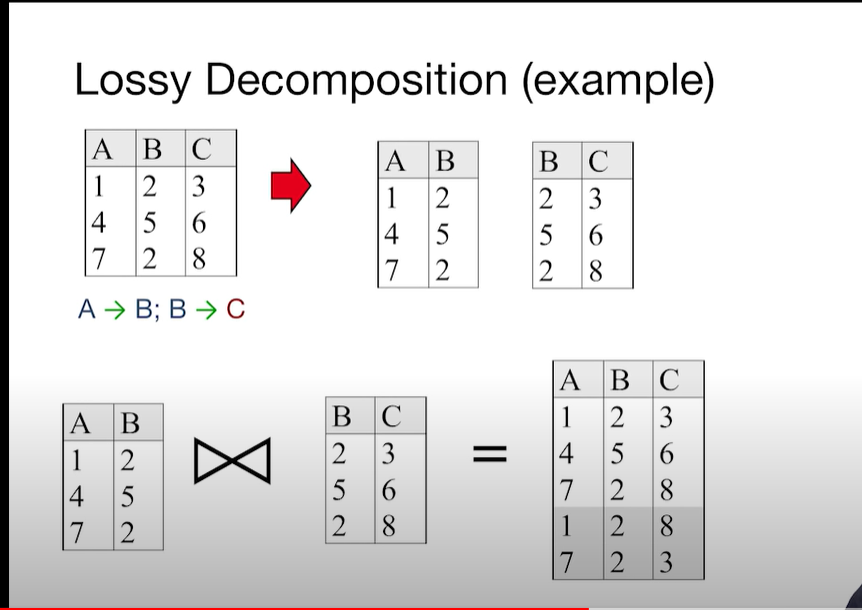
当两个链接与原来的relation r相等的时候,我们说他是lossless-join //成功的ddecomposition



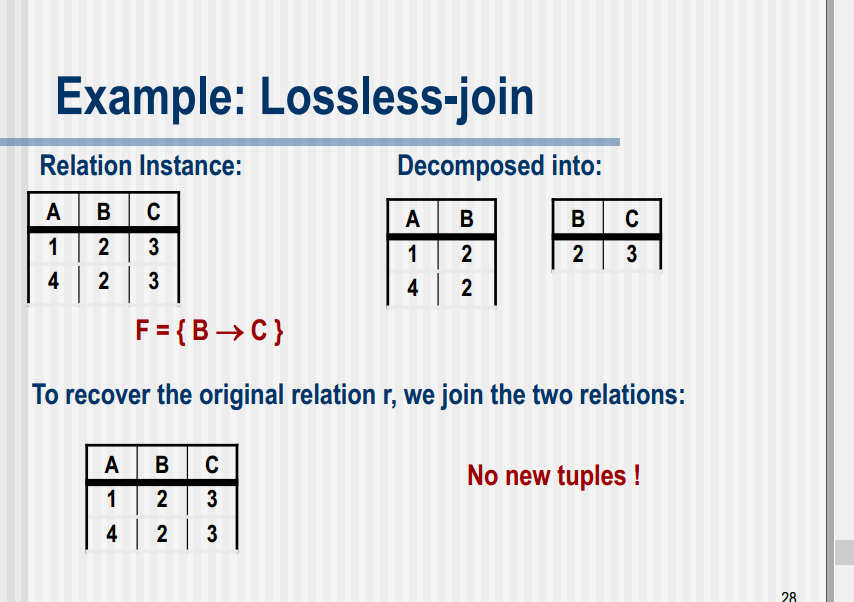
不管是好的坏的，r肯定是连接的子集，只有相等时Lossless-join

Theorem: 假设我们把relation schema R拆分成R1和R2，那么这个拆分是Lossless的当

，意思就是两边attribute的交集，必须是Xor Y 的Key//这样才能决定Xor Y



例如这里，交集是B，B不是B->C的KeY,因为B有重复了



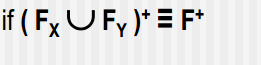
这里，交集是B，B是第二个relation的key，所以OK

Dependency-preserving decomposition

dependency-保存 分解

我们通过检查一个relation instance，来check是否保存了每一个FD

假设将R分成X与Y两组attribute， F是原来的FD们， FX是F关于X那组attribute的FD， FY是关于Y那组attribute的FD，

我们要确保， ，这样才能dependency-preserving// 就是XY交界处不应该有dependency，这样拆解的时候就不会损失FD