1.1. 50%. The rightmost page is always the one that is split, leaving two pages that are 50% full. The one on the left stays at 50%.

1.2. Instead of splitting 50/50, split 80/20. The page on the left is 80% full and won't be updated again (within that set of N inserts).

2.1. Candidates:
- a: lookup on x = 10, get rows, check y
- b: lookup on y between ..., get rows, check x
- c: Do both lookups, intersect, get rows.
- d: Index checks both conditions, get rows.
- e: Index cannot be used.
- f: Scan table, check both conditions.

So the possibilities, ranked are: d, c, (a, b), f

2.2. Candidates:
- a: --
- b: --
- c: Do both lookups, union, get rows.
- d: --
- e: --
- f: Scan table, check both conditions.

Ranked: c, f

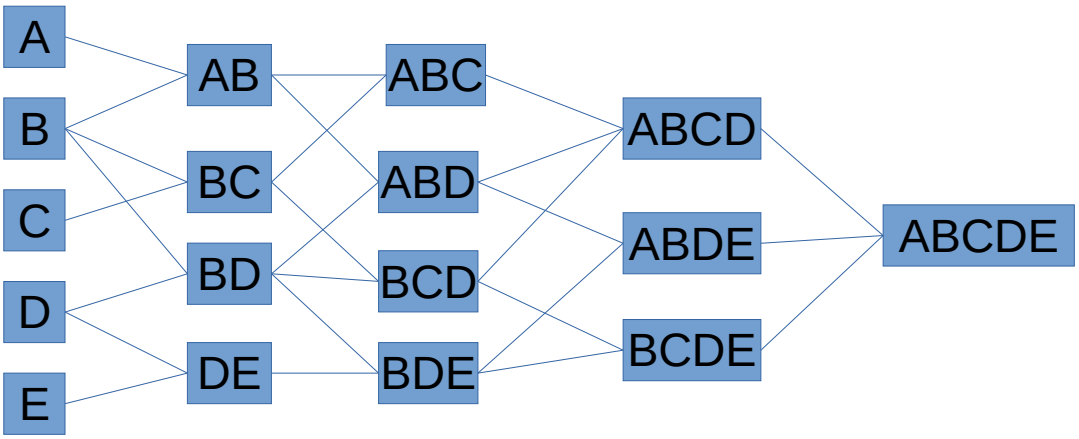
2.3
a: (y, x)
b: (y, z, x)
c: (x, y), (y, z), (z)

3.1.
```
union(X, Y):
  xset = {}
  for each x in X:
    - output x
    - add x to xset
  for each y in Y:
    if y not in xset:
      output y
```

This is faster because we don't have to save and then sort the Y values.

3.2. One pass over X, Y either way, but less memory needed if smaller input is scanned first.

4.1.



4.2. NOT considered:

j3: C and D are not connected by a foreign key or joined in the query.
j4: Not a left-deep join plan.
j6: AB and E are not connected by a foreigh key or joined in the query.
j8: Not a left-deep join plan.
j9: Not a left-deep join plan.

4.3.

a. Index on Rx

b. seq scan + filter -> index scan

c. T(tid) -> T(tid, z)
   S(sid) -> S(sid, y)

5.1. T1 < T2 < T3

5.2. T3 < T2 < T1

5.3. T2 < T3 < T1

5.4. T1 Sa should not have been granted. Not serializable:

- a: T1 < T2
- b: T2 < T3
- c: T3 < T1

which is cyclic

6.1: 3 versions:

- a = 1 (initial state)
- a = 2 (T2)
- a = 3 (T3)

6.2: 1

6.3: 1, 1, 1, 1, 1

6.4:

- a: 3
- b: 2
- c: 3
- d: 3
- e: 4

6.5: Block on update. Error (can't serialize) on OTHER commit.