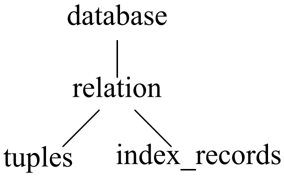
# CS386D Problem Set #7

[1] Consider the following three relations, each having 9 attributes listed in increasing order of selectivity (i.e., the most selective attribute is listed first):

```
relation A { type-a1 A1;          // type-a1 is the data type for attribute A1
             ...
             type-an A9;
           }
relation B { type-b1 B1;
             ...
             type-bn B9;
           }
```

For the following questions, assume the MGL graph below. Also, assume that the first five attributes of relations A and B are indexed. Using the *Degree 2 Cursor Stability Record Locking Protocol (CS2)* presented in class, what locks would be *taken and held* till the end of the transaction for each statement? And how many locks would this be? (Assume *n* is the number of tuples that fully qualify and let *m* be the number of tuples that are qualified only by index records).



```
(a) select *
    from   A
    where  A2= 2 and A4 = 4


(b) select *
    from   A
    where  A8 = 8


(c) update A
    set    A1 = a1, A2 = a2
    where  A8 = 9 and A5 = 5


(d) delete A
    where  A4 = 4


(e) select *
    from   A,B
    where  A3=3 and B4=4 and A2=B2


(f) update A
    set A1 = A1+1
    where A1 > 3
```

[2] Resolve [1], except use the RR2 protocol.

# Solutions

Note: in problem [1], I'm counting the number of locks that are held till the end of the transaction. The actual number of locks taken is a bit more — remember, in CS2 we place and remove locks on tuples we are not interested in. These locks are NOT counted below. And remember: attributes **A1..A5** and **B1..B5** are indexed; the remaining attributes of relations **A** and **B** are not indexed.

[2a] The following 3 + *n* locks are taken:

```
IR(database), IR(relation A), R(A2=2),
foreach tuple t that satisfies (A2=2) and (A4=4): R(t)
```

[2b] The following 2 locks would be held:

```
IR(database), R(relation A)
```

Note that the indices don't help in restricting the read set of **A**; hence a read lock on relation **A** is needed.

[2c] The following 5 + 3*n* locks are taken:

```
IW(database), IW(relation A), w(A1=a1), W(A2=a2), R(A5=5),
foreach tuple t that satisfies (A8=9) and (A5=5):
          W(t), W(A1=t[A1]), W(A2=t[A2])
```

Note that the index records (**A1=a1**) and (**A2=a2**) are write-locked because their index records are updated. The index lock (**A5=5**) is locked in read mode because the record is read, not updated. Each qualified tuple is write-locked for updates.

[2d] The following 3 + 5*n* locks are taken:

```
IW(database), IW(relation A), W(A4=4)
foreach tuple t that satisfies (A4=4):
      W(t), W(A1=t[A1]), W(A2=t[A2]), W(A3=t[A3]), W(A5=t[A5])
```

Note that each tuple to be deleted is write-locked, along with every index record that references that tuple is write-locked because those index records are to be updated.

[2e] *Note: locks are taken only during retrieval, NOT during join processing*. The following 5 + 2*n* locks are held:

```
IR(database), IR(relation A), R(A3=3),
foreach tuple t that satisfies (A3=3): R(t)
IR(relation B), R(B4=4),
foreach tuple r that satisfies (B4=4): R(r)
```

[2f] This is a tricky one. If you process this query using the **A1** index, you'll see that you'll fall into an endless loop. That is, you update a tuple, and its new index record will be inserted into the list of tuples that you have to update again. This is called the *Halloween Problem*. So, the only way to process this query is by using a scan. A total of 2 locks are taken, even though many **A** records and index records may be updated.

```
IW(database), W(relation A)
```

Note: problem 3 is the same as problem #2. Differences in the answers are (1) tuples are qualified only on index predicates (and not also on residuals) and (2) (basically) $n$ is replaced by $m$ in the number of locks taken, where $n<m$.

[3a] The following 3 + $m$ locks are taken:

```
IR(database), IR(relation A), R(A2=2),
foreach tuple t that satisfies (A2=2): R(t)
```

[3b]The following 2 locks are taken:

```
IR(database), R(relation A)
```

[3c] The following 5 + $m$ + 2$n$ locks are taken:

```
IW(database), IW(relation A), w(A1=a1), W(A2=a2), R(A5=5),
foreach tuple t that satisfies (A5=5): W(t)
      and for those that also satisfy (A8=9): W(A1=t[A1]), W(A2=t[A2])
```

Note here: all tuples that satisfy (`A5=5`) are locked in W mode. Only for those tuples that are actually updated (those that satisfy `A8=9`) will write locks be placed on `A1=t[A1]` and `A2=t[A2]`.

[3d] The following 3 + 5$m$ locks are taken:

```
IW(database), IW(relation A), W(A4=4)
foreach tuple t that satisfies (A4=4):
      W(t), W(A1=t[A1]), W(A2=t[A2]), W(A3=t[A3]), W(A5=t[A5])
```

[3e] The following 5 + 2$m$ locks are taken:

```
IR(database), IR(relation A), R(A3=3),
foreach tuple t that satisfies (A3=3): R(t)
IR(relation B), R(B4=4),
foreach tuple r that satisfies (B4=4): R(r)
```

[3f] The following 2 locks are taken:

```
IW(database), W(relation A)
```