UNIVERSITÄT DES SAARLANDES
Fachrichtung 6.2 – Informatik
Prof. Dr.-Ing. G. Weikum
Dipl.-Inform. R. Schenkel

6. Übung zur Vorlesung *Transaktionsorientierte Informationssysteme* im Sommersemester 2000

6.1. Discuss how predicate locking can be extended to disjunctive conditions such as queries of the form

```
SELECT Name FROM Emp
WHERE Position='Manager' OR Department='Research'.
```

Also discuss how join queries such as

```
SELECT Emp.Name, Emp.Department FROM Emp, Dept
WHERE Emp.Position='Manager' AND Dept.City='Toronto'
AND Emp.Department=Dept.Department
```

could be (conservatively) handled by predicate locking.

6.2. Consider the following transactions:

$t_1=r_1(x)\ w_1(x)\ r_1(y)\ w_1(y)$

$t_2=r_2(x)$

$t_3=r_3(y)\ w_3(y)$

Try to decompose t_1 into three pieces s.t. the result is a correct chopping.

6.3. Show that if a set of chopped transactions contains an *sc* cycle, any further chopping of any of the transactions will not render the graph acyclic.

6.4. Discuss how the incremental key-range locking protocol on index keys has to be changed for a unique index. Discuss also how unique and non-unique indexes may be treated differently by the various optimizations sketched in Section 9.5.

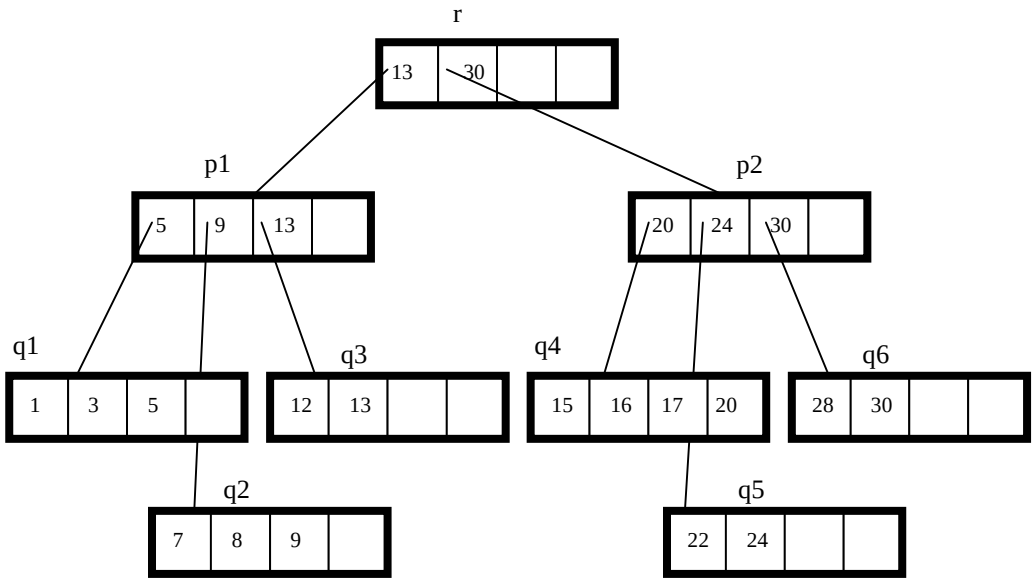
6.5. Incremental key-range locking based on read and write locks on index keys, as explained in Section 9.3, does not allow two insert operations in the same, previously unoccupied key interval to proceed concurrently. For example, in the index of Figure 9.5, an insert(27) would block a later insert(28) operation, although none of these keys are initially present and the two operations should be considered as commutative. Discuss whether the resulting blocking of one of the two operations can be safely avoided, and how this could be accomplished. Also discuss the analogous situation for delete operations, and the combination of insert and delete operations.

6.6. Consider a situation where a node *p* in a B^+-tree is not split-safe, but has a split-safe descendant *q*, with both *p* and *q* on the traversal path of an insert operation. Discuss how the lock-coupling technique should handle this situation to minimize lock contention.

6.7. Assume that the leaf nodes of a B^+-tree index are linked together bidirectionally (rather than only in forward direction). This may be useful to support range scans in descending key order, which in turn is of interest for returning qualifying records in descending sort order (as requested, for example, by a decision-support query). Discuss the consequences of this bi-directional linkage with regard to the lock-coupling and the link techniques for

the page-level concurrency control. If possible, generalize these techniques to accommodate the new feature.

6.8. Consider the following B^+-tree index on the attribute `AccountNumber` of an `Accounts` table. Assume that all tree nodes have a space capacity to hold up to four entries.



Which locks need to be requested for the execution of the following transaction, assuming the incremental key-range locking at the access layer, lock coupling at the page layer?

```
begin of transaction;
   select count(*) from Accounts where AccountNumber between 11 and 25;
   insert into Accounts (AccountNumber,...) values (27,...);
commit transaction;
```

6.9. Discuss to what extend lock escalation may incur deadlocks or livelocks.

6.10. Give examples for schedules that fall into the following five isolation-level classes (but not into the next larger, surrounding class): not even read uncommitted, read uncommitted, read committed, repeatable read, serializability.

6.11. Give an example for a snapshot-isolated history violating the consistency of the persistent data. Explain what kinds of inconsistencies can arise.

6.12. Design a concurrency control protocol that can guarantee snapshot isolation (and can indeed generate all snapshot-isolated histories).