Database Design and Implementation
Chapter 8 Conceptual Exercises

Questions	Answers
8.1.	The output will be an empty result set. In a product operation (also known as a cross join or cartesian product), each row in the first table is combined with every row in the second table. If one of the tables has no rows, then there won't be any combinations to produce, and the result set will be empty.

8.2.

```
// Creating the nodes for the STUDENT, DEPT, ENROLL, and SECTION tables
Layout studentLayout = mdm.getLayout("STUDENT", tx);
Scan studentScan = new TableScan(tx, "STUDENT", studentLayout);

Layout deptLayout = mdm.getLayout("DEPT", tx);
Scan deptScan = new TableScan(tx, "DEPT", deptLayout);

Layout enrollLayout = mdm.getLayout("ENROLL", tx);
Scan enrollScan = new TableScan(tx, "ENROLL", enrollLayout);

Layout sectionLayout = mdm.getLayout("SECTION", tx);
Scan sectionScan = new TableScan(tx, "SECTION", sectionLayout);

// Creating the Product nodes
Scan productStudentEnroll = new ProductScan(studentScan, enrollScan);
Scan productStudentEnrollDept = new ProductScan(productStudentEnroll,
deptScan);
Scan productAll = new ProductScan(productStudentEnrollDept, sectionScan);

// Creating the Select node
Predicate pred = new Predicate();
pred.addFieldMatch("SId", "StudentId");
pred.addFieldMatch("SectId", "SectionId");
pred.addFieldMatch("DId", "MajorId");
pred.addConstantMatch("YearOffered", 2020);
Scan finalScan = new SelectScan(productAll, pred);

// Outputting the results
while (finalScan.next()) {
   System.out.println(finalScan.getString("sname") + ", " +
                  finalScan.getString("dname") + ", " +
                  finalScan.getString("grade"));
}
finalScan.close();
```

Questions	Answers
8.3.a.	The transaction will need to obtain read locks (shared locks) on the **STUDENT** and **DEPT** tables in order to perform the table scans.

8.3.b.	If another transaction currently holds a write lock (exclusive lock) on the **STUDENT** or **DEPT** tables (maybe due to an update or delete operation), then the code will have to wait until that lock is released to acquire its read lock.
8.4.a.	If the first underlying scan has no records, then the **ProductScan** will never progress to scan the records of the second underlying scan. The fix would be to check whether the first scan has records and if it doesn't, skip the **ProductScan** entirely or move on to the next operation.
8.4.b.	The second scan is effectively nested within the first scan. For each record of the first scan, the second scan iterates over all its records. If the second scan has no records, the first scan will still be iterated, but for each iteration of the first scan, the second will immediately report that there are no records to combine with.
8.5.a.	The issue with using the same table scan twice in the product is that both instances of the scan will be pointing to the same position in the table. When you advance the scan in one instance, the other instance will also move forward. This will produce strange results as the product will only combine a student record with itself, rather than producing all pairs.
8.5.b.	The problem with this approach is redundancy. You will get pairs like (StudentA, StudentB) and (StudentB, StudentA) which represent the same combination but in a different order. This results in twice the number of combinations than required.