$$\begin{aligned} Total\ Cost &= (\#\ CPU sorting) * (sort\ cost) \\ &= 100 * (3 * 1,000 * t_d) \\ &= 300,000 * t_d \end{aligned}$$

$$Elapsed\ Time = 1,000\ pgs * (sort\ cost)$$

Exercise 21.6 Consider the Employees and Departments relations described in Exercise 21.3. They are now stored in a distributed DBMS with all of Employees stored at Naples and all of Departments stored at Berlin. There are no indexes on these relations. The cost of various operations is as described in Exercise 21.3. Consider the query:

```
SELECT *
FROM   Employees E, Departments D
WHERE  E.eid = D.mgrid
```

The query is posed at Delhi, and you are told that only 1 percent of employees are managers. Find the cost of answering this query using each of the following plans:

1. Compute the query at Naples by shipping Departments to Naples; then ship the result to Delhi.
2. Compute the query at Berlin by shipping Employees to Berlin; then ship the result to Delhi.
3. Compute the query at Delhi by shipping both relations to Delhi.
4. Compute the query at Naples using Bloomjoin; then ship the result to Delhi.
5. Compute the query at Berlin using Bloomjoin; then ship the result to Delhi.
6. Compute the query at Naples using Semijoin; then ship the result to Delhi.
7. Compute the query at Berlin using Semijoin; then ship the result to Delhi.

Answer 21.6 First let us calculate N, the number of pages required to hold the result of the query. The query involves an equijoin on the *eid* field, which is a key for the Employees relation. Hence, there will be exactly one result tuple for each Departments tuple. Now, each page can hold 4000 bytes, and each Departments tuple is 20 bytes long. So 200 tuples fit on a page, and there are 5000 pages of Departments records. Thus, we have $5000 * 200 = 10^6$ result tuples. Now, each result tuple (join of Employees and Departments) is 40 bytes long; 100 of these result tuples will fit on a page holding 4000 bytes. So $N = \frac{10^6}{100} = 10000$ pages.

In the following, we use Sort-Merge Join to compute any join, the cost of which is $3*(M+N)$, where M, N are the number of pages of the two relations being joined.

1.

$$\begin{aligned} TotalCost &= Shipping\ Departments\ Berlin \longrightarrow Naples = 5000t_s \\ &+ Cost\ of\ computing\ query\ at\ Naples = 3 * (100,0000 + 5000)t_d \\ &+ Shipping\ result\ Naples \longrightarrow Delhi = N * t_s \\ &= 5000t_s + 315,000t_d + 10,000 * t_s \\ &= 15,000t_s + 315,000t_d \end{aligned}$$

2.

$$\begin{aligned} Total\ Cost &= Shipping\ Employees\ Naples \longrightarrow Berlin = 100,000t_s \\ &+ Cost\ of\ computing\ query\ at\ Berlin = 3 * (100,0000 + 5000)t_d \\ &+ Shipping\ result\ Berlin \longrightarrow Delhi = N * t_s \\ &= 100,000t_s + 315,000t_d + 10,000 * t_s \\ &= 110,000t_s + 315,000t_d \end{aligned}$$

3.

$$\begin{aligned} Total\ Cost &= Shipping\ Employees\ Naples \longrightarrow Delhi = 100,000t_s \\ &+ Shipping\ Departments\ Berlin \longrightarrow Delhi = 5000t_s \\ &+ Cost\ of\ computing\ query\ at\ Delhi = 3 * (100,0000 + 5000)t_d \\ &= 100,000t_s + 5000t_s + 315,000t_d \\ &= 105,000t_s + 315,000t_d \end{aligned}$$

4. We need to calculcate the cost of Bloomjoin at Naples.
   The plan is to calculate the bit-vector (corresponding to Employees) at Naples, then ship the bit-vector to Berlin, calculate the reduction of Departments at Berli, ship the reduction to Naples, calculate the join at Naples, and (finally!) ship the result to Delhi.

$$\begin{aligned} Total\ Cost &= Hashing\ Employees\ at\ Naples = 100,000t_d \\ &+ Shipping\ bit-vector\ Naples \longrightarrow Berlin = 5000t_s? \\ &+ Reduction\ of\ Departments\ at\ Berlin = 5000t_d \\ &+ Shipping\ Reduction\ of\ Departments\ Berlin \longrightarrow\ Naples = 5000t_s \\ &+ Computing\ join\ at\ Naples = 3 * (100,000 + 5000)t_d \\ &+ Shipping\ result\ Naples \longrightarrow Delhi = N * t_s \\ &= 100,000t_d + 5000t_s? + 5000t_d + 5000t_s + 315,000t_d + 10,000t_s \\ &= 420,000t_d + 20,000t_s? \end{aligned}$$

5. We need to calculcate the cost of Bloomjoin at Berlin.

$$\begin{aligned} Total\ Cost &= Hashing\ Departments\ at\ Berlin = 5000t_d \\ &+ Shipping\ bit-vector\ Berlin \longrightarrow Naples = 250t_s? \end{aligned}$$

$$
\begin{aligned}
&+ \quad \textit{Reduction of Employeesat Naples} = 100,000t_d \\
&+ \quad \textit{Shipping Reduction of Employees Naples} \longrightarrow \textit{Berlin} = 100t_s? \\
&+ \quad \textit{Computing join at Berlin} = 3 * (5000 + 100)t_d? \\
&+ \quad \textit{Shipping result Berlin} \longrightarrow \textit{Delhi} = N * t_s \\
&+ \quad 5000t_d + 250t_s? + 100,000t_d + 100t_s? + 15,300t_d? + 10,000t_s \\
&+ \quad 120,300t_d + 10,350t_s??
\end{aligned}
$$

6. We need to calculcate the cost of Semijoin at Naples.

   The plan is to project the *eid* field of Employees at Naples, then ship the projection to Berlin, calculate the reduction of Departments w.r.t. Employees at Berlin, ship the reduction to Naples, calculate the join at Naples, and ship the result to Delhi.

   Let us assume the size of the *eid* field is 10 bytes. Cost of projecting the *eid* field is $100,000t_d$ for the scan of the Employees relation, and $50,000t_d$ for creating a temporary file. (Note the *eid* field is half the length of an Employess record.) If the optimizer to smart to recognize that *eid* is a key field, it will not try to eliminate duplicates. Else the projection will incur additional cost in sorting, and then scanning to eliminate duplicates. For our purposes, let us assume we have a smart optimizer.

$$
\begin{aligned}
\textit{Total Cost} &= \textit{Projecting Employees at Naples} = 150,000t_d \\
&+ \quad \textit{Shipping projection Naples} \longrightarrow \textit{Berlin} = 50,000t_s \\
&+ \quad \textit{Reduction of Departments at Berlin} = 3 * (50,000 + 5000)t_d \\
&+ \quad \textit{Shipping Reduction of Departments Berlin} \longrightarrow \textit{Naples} = 5000t_s \\
&+ \quad \textit{Computing join at Naples} = 3 * (100,000 + 5000)t_d \\
&+ \quad \textit{Shipping result Naples} \longrightarrow \textit{Delhi} = N * t_s \\
&= \quad 150,00t_d + 50,000t_s + 165,000t_d + 5000t_s + 315,000t_d + 10,000t_s \\
&= \quad 630,000t_d + 65,000t_s
\end{aligned}
$$

7. We need to calculcate the cost of Semijoin at Berlin.

   The plan is to project the *mgrid* field of Departments at Berlin, then ship the projection to Naples, calculate the reduction of Employeess w.r.t. Departments at Naples, ship the reduction to Berlin, calculate the join at Berlin, and ship the result to Delhi.

   Let us assume the size of the *mgrid* field is 10 bytes. Cost of projecting the *mgrid* field is $5000t_d$ for the scan of the Employees relation, and $2500t_d$ for creating a temporary file. (Note the *mgrid* field is half the length of a Departments record.) Now the *mgrid* field is not a key field, and we need to eliminate duplicates as part of the projection.

$$
\begin{aligned}
\textit{Total Cost} &= \textit{Projecting Departments at Berlin} \\
&+ \quad \textit{Shipping projection Berlin} \longrightarrow \textit{Naples} = 2500t_s \\
&+ \quad \textit{Reduction of Employees at Naples} = 3 * (100,000 + 2500)t_d \\
&+ \quad \textit{Shipping Reduction of Employees Naples} \longrightarrow \textit{Berlin} = 1000t_s \\
&+ \quad \textit{Computing join at Berlin} = 3 * (5000 + 1000)t_d
\end{aligned}
$$

$$
\begin{aligned}
&+ \quad \textit{Shipping result Berlin} \longrightarrow \textit{Delhi} = N * t_s \\
&= \quad t_d + 2500t_s + 307,500t_d + 1000t_s + 18,000t_d + 10,000t_s \\
&= \quad t_d + 13,500t_s
\end{aligned}
$$

Exercise 21.7 Consider your answers in Exercise 21.6. Which plan minimizes shipping costs? Is it necessarily the cheapest plan? Which do you expect to be the cheapest?

Answer 21.7 Answer not available.

Exercise 21.8 Consider the Employees and Departments relations described in Exercise 21.3. They are now stored in a distributed DBMS with 10 sites. The Departments tuples are horizontally partitioned across the 10 sites by *did*, with the same number of tuples assigned to each site and with no particular order to how tuples are assigned to sites. The Employees tuples are similarly partitioned, by *sal* ranges, with $sal \leq 100,000$ assigned to the first site, $100,000 < sal \leq 200,000$ assigned to the second site, and so on. In addition, the partition $sal \leq 100,000$ is frequently accessed and infrequently updated, and it is therefore replicated at every site. No other Employees partition is replicated.

1. Describe the best plan (unless a plan is specified) and give its cost:
   (a) Compute the natural join of Employees and Departments using the strategy of shipping all fragments of the smaller relation to every site containing tuples of the larger relation.
   (b) Find the highest paid employee.
   (c) Find the highest paid employee with salary less than $100,000$.
   (d) Find the highest paid employee with salary greater than $400,000$ and less than $500,000$.
   (e) Find the highest paid employee with salary greater than $450,000$ and less than $550,000$.
   (f) Find the highest paid manager for those departments stored at the query site.
   (g) Find the highest paid manager.
2. Assuming the same data distribution, describe the sites visited and the locks obtained for the following update transactions, assuming that *synchronous* replication is used for the replication of Employees tuples with $sal \leq 100,000$:
   (a) Give employees with salary less than $100,000$ a 10 percent raise, with a maximum salary of $100,000$ (i.e., the raise cannot increase the salary to more than $100,000$).
   (b) Give all employees a 10 percent raise. The conditions of the original partitioning of Employees must still be satisfied after the update.
3. Assuming the same data distribution, describe the sites visited and the locks obtained for the following update transactions, assuming that *asynchronous* replication is used for the replication of Employees tuples with $sal \leq 100,000$.
   (a) For all employees with salary less than $100,000$ give them a 10 percent raise, with a maximum salary of $100,000$.