Analysis of Query Processing Performance: An Example

In the following, assume the latency/transfer-rate model of disk performance, where we estimate disk access times by allowing blocks that are consecutive on disk to be fetched with a single seek time and rotational latency cost (as shown in class). Also, we use the term RID (Record ID) to refer to an 8-byte logical pointer that can be used to locate a record (tuple) in a table.

You are given the following online movie rental database schema. (Note that in this schema, there are no copies of movies – we only keep track of who rented each movie, where a rental means viewing a movie online.)

```
Customer (cname, cid, address, city, phone)
Movie (title, genre, date_released, general_description)
Rental (cid, title, date_rented, rental_cost)
```

Assume that there are 400,000 different movies, 10 million customers, and 2 billion rental records. Each tuple in Rental is of size 80 bytes, while all other tuples are of size 200 bytes each. Consider the following four queries on the given schema:

SELECT C.cname
FROM Customer C, Rental R
WHERE C.cid= R.cid and R.title = “Rush Hour 3” and R.date_rented during ”Nov 22, 2007”

SELECT C.cname, M.title
FROM Customer C, Movie M, Rental R
WHERE C.cid = R.cid and R.title = M.title and M.genre = ”Action” and M.date_release = 2007

SELECT M.title
FROM Movie M, Rental R
WHERE M.title = R.title and R.date_rented - M.date_released $<$ 180 days

SELECT C.cname, M.title
FROM Customer C, Movie M, Rental R
WHERE C.cid = R.cid and R.title = M.title and M.genre = ”Action” and M.date_release = 2007
and C.city = “Oakville”

(a) For each query, describe in one sentence what it does. (That is, what task does it perform?) Please assume that all the queries are correct even though we use some simplified syntax for dates.

In the following, assume that each movie has the same number of rentals, i.e., each movie has been watched 5000 times. To describe how a query is executed, draw a query plan tree and describe in words what algorithms should be used for the various selection and joins. Also provide estimates of the running times (we assume the times are dominated by disk accesses).

(b) Assume that there are no indexes on any of the relations, and that all relations are unclustered (not sorted in any way). Describe how a database system would best execute the four queries in this case, given that 200MB of main memory are available for query processing, and assuming a hard disk with 8 ms for seek time plus rotational latency (i.e., a random access requires 8 ms to find the right position on disk) and a maximum transfer rate of 40 MB/s.

Assume that 5% of all movies are action movies, 20% of all action movies were released in 2007, 2% of all movies were released in 2007, 0.2% of all rentals were done on Nov 22, 2007, 30% of all rentals were made within 180 days of the release date, and 2% of all rentals of ”Rush Hour 3” were done on Nov 22, 2007. Also, there are 20 customers living in Oakville.

(c) Consider a sparse clustered B^+-tree index on cid in the Customer table, and a dense unclustered B^+-tree index on cid in the Rental table, where attribute cid has a size of 12 bytes. For each index, what is the height and the size of the tree? How long does it take to fetch a record with a particular cid using these indexes?

(d) Now assume that there are unclustered indexes on the title, date_released, and genre attributes of Movie, and on the cname attribute of Customer. Also assume that there is a sparse clustered index on the date_rented attribute of Rental. Describe how a database system would now best execute the **first three queries**, under the same assumptions as in (b). Which indexes would you use? Estimate the running times. (You may estimate the heights of the indexes, so there is no need to repeat the analysis in (c) for each index.)

(e) Suppose it is very important to optimize the performance of the second query above. What selection of index structures would you create to achieve this, and what would be the resulting running time?

(f) And how about the fourth query? What is the best way to execute that query if you could build any index you want?