Quiz 10: Physical DB Design and NoSQL

Initial Score (out of 10)

Taken by: _____________________________________________

Name

Student ID

● We will discuss the answers right after the quiz.
● You are to self-grade and record your actual initial score (above) as we do so.
● You should also record the correct answers for any problems that you miss.
● A full 10 points for taking this quiz will be included as part of your quiz grade.

1. (5 points) Consider the following schema for an e-commerce application:

Diver(\textit{did}, \textit{firstname}, \textit{lastname}, \textit{age}, \textit{expertise\_level})

\textbf{Purchases}(\textit{did, eid, date})

\textbf{Equipment}(\textit{eid, name,price})

Come up with a good physical database design to support a workload that consists of the following expected mix of queries:

\textbf{Q0:} \texttt{SELECT * FROM Equipment WHERE \textit{eid} = value1;}

\textbf{Q1:} \texttt{SELECT * FROM Diver WHERE firstname LIKE \textquote{value1\%};}

\textbf{Q2:} \texttt{SELECT \textit{price, COUNT(*) FROM Equipment GROUP BY \textit{price};}

\textbf{Q3:} \texttt{SELECT \textit{E.name, E.price FROM Equipment E, Purchases P, Diver D WHERE D.age = value1 AND E.eid = P.eid AND D.did = P.did;}

\textbf{Q4:} \texttt{SELECT * FROM Equipment WHERE name LIKE \textquote{value1\%}}

Suppose this data will be residing in a DBMS that only offers B+ Trees as its choice of index type. Come up with a good set of indexes for the data and the workload above – indicating, for each recommended index, the indexed columns and the motivating queries in the table below. Use the first column of the table to number the indexes from 1 to N for quick reference in subsequent parts of this problem. (Not all rows need to be filled in.)

<table>
<thead>
<tr>
<th>Index Number</th>
<th>Indexed Table</th>
<th>Indexed Column(s)</th>
<th>Motivating Query(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Equipment</td>
<td>\textit{eid}</td>
<td>Q0, Q3</td>
</tr>
<tr>
<td>2</td>
<td>Equipment</td>
<td>\textit{price}</td>
<td>Q2</td>
</tr>
<tr>
<td>3</td>
<td>Purchases</td>
<td>(\textit{did,eid}) or \textit{did}</td>
<td>Q3</td>
</tr>
<tr>
<td>4</td>
<td>Diver</td>
<td>\textit{age or (age,did)}</td>
<td>Q3</td>
</tr>
<tr>
<td>5</td>
<td>Equipment</td>
<td>\textit{name}</td>
<td>Q4</td>
</tr>
</tbody>
</table>
(2) (3 points) Your design above should have at least one entry for each of the database's tables. Indicate below (by index number) what are your index option(s) to be considered for a clustered index, if any, and briefly say why.

(i) Clustered index(es) for Diver: age(clustered) - (age, did) Unclustered

(ii) Clustered index(es) for Purchases: did (clustered) - (did, eid) Unclustered

(iii) Clustered index(es) for Equipment: 5
    (2 is index only)(1 : in Q0 it is primary key lookup, at max one match).

3. (2 points) In NOSQL world, instead of having three relations, we could insert a bag of (eid, date) pairs in Diver or a bag of (did, date) in Equipment table. Briefly mention why keeping the relations as it is (relational case) would be better for this example than any of the mentioned NOSQL alternatives.

Disadvantage: Objects will get large by each purchase in an ever-growing fashion. Also, it needs indexing on nested data in order to go in the other direction.