1 Buffer Management

(a) Fill in the following tables for the given buffer replacement policies. You have 4 buffer pages, with the access pattern A B C D A F A D G D G E D F
(b) Fill in the following tables for the given buffer replacement policies. You have 4 buffer pages, with the access pattern A, B, C, D, A, F (remains pinned), D, G, D, unpin F, G, E, D, F. Remember that unpinning does not contribute to the hit count!

**Least Recently Used w/ Pinning**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>*</th>
<th></th>
<th></th>
<th></th>
<th>E</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>F</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>G</td>
<td>*</td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

6

13

**Most Recently Used w/ Pinning**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>*</th>
<th>F</th>
<th>x</th>
<th>x</th>
<th>x</th>
<th>x</th>
<th>G</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>F</td>
</tr>
</tbody>
</table>

3

13

* = hit
X = pinned

Remember that unpinning doesn’t contribute to the hit count.

(c) Is MRU ever better than LRU?
Yes; MRU prevents sequential flooding during sequential scans.

(d) Why would we use a clock replacement policy over LRU?
Efficiency (approximation of LRU; don’t need to maintain entire ordering).

(e) Why would it be useful for a database management system to implement its own buffer replacement policy? Why shouldn’t we just rely on the operating system?
The database management system knows its data access patterns, which allows it to optimize its buffer replacement policy for each case.
2 Relational Algebra

Why do we care about relational algebra? Why do you think it might be useful?

Relational algebra are plans to execute queries; the many ways of writing the plans give the system room to design for optimizations. We will learn more about how to estimate the cost of the plan in the future.

Consider the schema:

\[
\begin{align*}
\text{Songs} & \quad \text{(song_id, song_name, album_id, weeks_in_top_40)} \\
\text{Artists} & \quad \text{(artist_id, artist_name, first_year_active)} \\
\text{Albums} & \quad \text{(album_id, album_name, artist_id, year_released, genre)}
\end{align*}
\]

Write relational algebra expressions for the following queries:

(a) Find the name of the artists who have albums with a genre of either 'pop' or 'rock'.

\[
\pi \text{artists.artist_name} (\text{Artists} \bowtie (\sigma \text{albums.genre} = \text{'pop'} \lor \text{albums.color} = \text{'rock'} \text{Albums}))
\]

(b) Find the name of the artists who have albums of genre 'pop' and 'rock'.

\[
\begin{align*}
\pi \text{artists.artist_name} & \quad (\sigma \text{albums.genre} = \text{'pop'} \text{Albums}) \bowtie \text{Artists} \\
\pi \text{artists.artist_name} & \quad (\sigma \text{albums.genre} = \text{'rock'} \text{Albums}) \bowtie \text{Artists}
\end{align*}
\]

(c) Find the id of the artists who have albums of genre 'pop' or have spent over 10 weeks in the top 40.

\[
\begin{align*}
\pi \text{artists.artist_id} & \quad (\sigma \text{albums.genre} = \text{'pop'} \text{Albums}) \cup \pi \text{albums.artist_id} \\
& \quad (\sigma \text{songs.weeks_in_top_40} > 10 \text{Songs})
\end{align*}
\]

(d) Find the names of all artists who do not have any albums.

\[
\pi \text{artists.artist_name} (\text{Artists} \setminus (\pi \text{artists.artist_id} \bowtie \text{Artists}) - (\pi \text{albums.artist_id} \bowtie \text{Albums}))
\]