CS W186 Introduction to Database Systems
Spring 2020 Josh Hug, Michael Ball

DIS 4

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**

Least Recently Used (LRU)

													Hit Rate
A				*		*						F	$\frac{6}{14}$
	B				F						E		
		C						G		*			
			D				*		*			*	

Most Recently Used (MRU)

													Hit Rate
A				*	F	A							$\frac{2}{14}$
	B												
		C											
			D				*	G	D	G	E	D	F

Clock ("second chance LRU")

													Hit Rate
A				*	F						D		$\frac{4}{14}$
	B					A						F	
		C						G		*			
			D				*		*		E		

Red frame outline is where the clock hand points at the end of the access
Yellow frame outline is where the clock hand considered ejecting a page, but the bit was one
Red box means the second chance bit is zero
Green box means the second chance bit is one
* means a hit

(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

A				*							E		
	B				F	x	x	x	x	x			*
		C						G			*		
			D				*		*			*	

$\frac{6}{13}$

Most Recently Used w/ Pinning

A				*	F	x	x	x	x	x	G	E	
	B												
		C											
			D				*	G	D			*	F

$\frac{3}{13}$

* = 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:

```
Songs (song_id, song_name, album_id, weeks_in_top_40)
Artists(artist_id, artist_name, first_year_active)
Albums (album_id, album_name, artist_id,year_released,genre)
```

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'.

π artists.artist_name (Artists $\bowtie$(σ albums.genre = 'pop' $\vee$ albums.color = 'rock' Albums))

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

π artists.artist_name ((σ albums.genre = 'pop' Albums) $\bowtie$Artists) $\cap$
π artists.artist_name ((σ albums.genre = 'rock' Albums) $\bowtie$Artists)

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

π artists.artist_id (Artists $\bowtie$(σ albums.genre = 'pop' Albums)) $\cup\pi$ albums.artist_id (Albums $\bowtie$(σ songs.weeks_in_top_40 > 10 Songs))

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

π artists.artist_name (Artists $\bowtie$((π artists.artist_id Artists) - (π albums.artist_id Albums)))