Warmup #1  (Textbook Problem 8.21 (modified))

Normalize the following schema, with given constraints, to BCNF.

books(accessionno, isbn, title, author, publisher)
users(userid, name, deptid, deptname)

Accessionno → isbn
isbn → title
isbn → publisher
isbn → author
userid → name
userid → deptid
deptid → deptname

Answer:
books(isbn, title, publisher, author)

accession(accessionno, isbn)

users(userid, name, deptid)

departments(deptid, deptname)

Warmup #2  (Textbook Problem 10.11)

How does the remapping of bad sectors by disk controllers affect data-retrieval rates? Answer: Remapping of bad sectors by disk controllers reduces data retrieval rates because of the loss of sequentiality amongst the sectors.

Warmup #3  (Textbook Problem 10.15)

Explain why the allocation of records to blocks affects database-system performance significantly. Answer: If we allocate related records to blocks, we can often retrieve most, or all, of the requested records by a query with one disk access. Disk accesses tend to be the bottlenecks in databases; since this allocation strategy reduces the number of disk accesses for a given operation, it significantly improves performance.
Warmup #4  (Textbook Problem 11.3)

Construct a B+-tree for the following set of key values:

\[(2, 3, 5, 7, 11, 17, 19, 23, 29, 31)\]

Assume that the tree is initially empty and values are added in ascending order. Construct B+-trees for the cases where the number of pointers that will fit in one node is as follows:

a. Four
b. Six
c. Eight

Answer: The following were generated by inserting values into the B+-tree in ascending order. A node (other than the root) was never allowed to have fewer than \(\lceil n/2 \rceil\) values/pointers.

Problem 5 (To Be Graded)

Following is the schema for the database of music festivals:

- musician(mid, name, address)
- musician_in_band(mid, bid, instrument)
• band(bid, bandname, home_state, home_zip)

(1) Assume the following requirement for band: a state will contain multiple zip codes, but a zip code will always be in the same state. Different bands may have the same name, come from the same home state and have the same home zip code. Please answer the following questions:

(1.a) Is the band schema in BCNF? If it is not, decompose it into a schema that is in BCNF. Please ensure your decomposition to be lossless.

Answer: The band schema is not in BCNF. The decomposed schema is

\[
\begin{align*}
\text{band}(\text{bid}, \text{bandname}, \text{home_zip}) \\
\text{address}(\text{home_state}, \text{home_zip})
\end{align*}
\]

(1.b) Consider the following decomposition of band:

• band_name(bid, bandname)
• band_address(bandname, home_state, home_zip)

Is the above decomposition lossless? Is the result in BCNF? Please explain.

Answer:

• The above decomposition is not lossless. The functional dependencies

\[
\begin{align*}
\text{band_name} \cap \text{band_address} \rightarrow \text{band_name} \Rightarrow \text{bandname} \rightarrow \text{bid} \\
\text{band_name} \cap \text{band_address} \rightarrow \text{band_address} \Rightarrow \text{bandname} \rightarrow \text{home_state} \text{ home_zip}
\end{align*}
\]

are not part of the closure of the original set of functional dependencies

\[
\begin{align*}
\text{bid} \rightarrow \text{bandname} \text{ home_state} \text{ home_zip} \\
\text{home_zip} \rightarrow \text{home_state}
\end{align*}
\]

• The resulting schema of the above decomposition is not in BCNF, because \text{home_zip} is not a superkey in \text{band_address} in the following functional dependency

\[
\text{home_zip} \rightarrow \text{home_state}
\]

The reason why \text{home_zip} is not a superkey in \text{band_address} is because the different bands may have the same \text{bandname}, \text{home_zip} and \text{home_state}. An equivalent argument is that, in the functional dependency

\[\text{home_zip} \rightarrow \text{home_state}\]

The closure of \text{home_zip} does not include all attributes in \text{band_address}.

(2) Suppose a new entity set is added

• shows(sid, bid, ticket_price, qty_sold, total_profit)

Where:

• \text{total_profit} is the total amount of money that the show generated
• \text{qty_sold} refers to the number of tickets sold at the given ticket_price (each show only has a single ticket_price initially).
• Every show contains exactly one band, but bands may play in multiple shows.
Assume the following requirement: a show is capable of selling tickets for different seats at a venue, each with a different ticket_price. The qty_sold attribute represents the number of tickets sold of one specific ticket_price. A show will still only feature a single band. After adding these constraints, the shows table contains the following dependencies:

\[
\begin{align*}
\text{sid} & \rightarrow \text{bid} \\
\text{sid, ticket_price} & \rightarrow \text{qty_sold} \\
\text{sid} & \rightarrow \text{total_profit}
\end{align*}
\]

Please answer the following questions:
(2.a) The schema for the shows table is not in BCNF. Please explain why not.
Answer: Shows is not in BCNF because sid is not a superkey in shows and it is in the following functional dependencies

\[
\begin{align*}
\text{sid} & \rightarrow \text{bid} \\
\text{sid} & \rightarrow \text{total_profit}
\end{align*}
\]

An equivalent argument is that the closure of the functional dependencies is

\[
\text{sid} \rightarrow \text{bid total_profit}
\]

\[
\text{sid ticket_price} \rightarrow \text{sid bid total_profit ticket_price qty_sold}
\]

But the closure of sid does not include all attributes in shows. (2.b) Decompose shows into a new schema that is in BCNF. List all foreign keys on the final schema.
Answer: The result of decomposition is

\[
\begin{align*}
\text{shows(sid, bid, total_profit)} \\
\text{show_tickets(sid, ticket_price, qty_sold)}
\end{align*}
\]

The foreign keys are sid in show_tickets, and bid in shows.

Problem 6 (To Be Graded)

NOTE: All students were given full points for 6.2a since we provided the wrong functions to calculate the height of the tree. The height of the tree should be calculated using ceiling (rounding up) instead of floor (rounding down). However, problem 6.2b was graded regularly. If a student performed the proper calculations to get the number of disk reads with the height they calculated, then full for 6.2b was given.
Imagine that you have the following table:
The table contains 1024 rows in total. Answer the following questions about this table:

1. Suppose the database administrator decides to store \textit{bandId} as an 8-byte integer, \textit{bandName} as a 48-byte character array, and \textit{ranking} as a 8-byte integer. If an instance of \textit{bandName} is shorter than 48 bytes, the empty space will be filled with null characters.

   - All attributes of a tuple are stored in contiguous space within the same disk block\(^1\).
   - A disk block size is 512 bytes.
   - The disk on average performs the sequential read at 1ms per disk block and the random read at 10ms per disk block.

   (a) What is the maximum number of tuples that can be stored in a disk block? Answer: The maximum number of tuples can be stored in a disk block is

   \[
   \frac{512B}{8B + 48B + 8B} = 8
   \]

   (b) What is the minimum number of disk blocks that need to be allocated for all tuples in the table? Answer: The minimum number of disk blocks that need to be allocated for all tuples in the table is

   \[
   \frac{1024 \text{ tuples}}{8 \text{ tuples per block}} = 128 \text{ blocks}
   \]

   (c) What is the minimum time to read all tuples (in no particular order), assuming that the minimum number of disk blocks are allocated? Answer: The minimum time to read all tuples is when disk blocks are allocated next to each other. One random read incurs at the first block, and all sequential reads for the rest blocks thereafter.

   \[
   10ms \times 1 + 1ms \times (128 - 1) = 137ms
   \]

2. Suppose that a secondary index of B+-tree is created on the ranking column (which you can assume is a candidate key with unique values). Assume the following

\(^1\)The minimum storage unit on a disk is called a \textit{disk block} or a \textit{disk page}. 
• Each tree node has a size of 64 bytes.
• The data in a tree node are stored in contiguous space within the same disk block.
• The tree has a fanout of 4 and each leaf node is 60% full\(^2\).
• A pointer is 8-byte long.

(a) How many bytes of space does the secondary index require? Answer: The number of entries in a node is
\[
\left\lfloor \frac{64 \text{ bytes per node} \times 60\%}{8 \text{ bytes per ranking} + 8 \text{ bytes per pointer}} \right\rfloor = 2
\]
The number of leaf nodes is
\[
\frac{1024 \text{ rankings}}{2 \text{ rankings per node}} = 512
\]
The height of the tree is
\[
\left\lceil \log_{\text{fanout}}(\text{num.leaves}) \right\rceil = \left\lceil \log_4(512) \right\rceil = 5
\]
The total number of nodes is
\[
\frac{\text{fanout}^{\text{height} + 1} - 1}{\text{fanout} - 1} = \frac{4^{5+1} - 1}{4 - 1} = 1365
\]
The total size of the tree is
\[
\text{node size} \times \text{num.nodes} = 64B \times 1365 = 87360B
\]

(b) How many disk reads (including index search and tuple retrieval) in the worst case are required to find a tuple via a specific ranking? Answer: The height of the tree is
\[
\left\lceil \log_{\text{fanout}}(\text{num.leaves}) \right\rceil = \left\lceil \log_4(512) \right\rceil = 5
\]
In the worst case, the number of disk reads is equal to the number of nodes visited plus one disk read to retrieve the tuple, or
\[
(1 + \text{tree height}) + (1 \text{ tuple read}) = 1 + 5 + 1 = 7
\]

\(^2\)The ratio of the number entries over the maximum number of entries is at least 60%.