Week 8: Worksheet
CSC 343 Winter 2021
University of Toronto Mississauga

March 18/19th, 2021

Recap

Materialized View

Materialized Views (psql documentation) are logical views (think tables) of our data, driven by a SELECT query that results in the data queried being stored in the database. This is beneficial because the data is stored and, thus, indexed! So it can be faster to JOIN this type of view with other tables.

Virtual View

Virtual Views (aka just 'views', psql documentation) are logical *virtual* views (think tables stored in memory) not stored anywhere on disk, so each time needed they must be generated. This allows for the most up-to-date information based on our dataset.

M.V. vs. V.V.

1. V.V. is stored in memory, while M.V. is stored on disk. *Note: this means that M.V. have an innate cost (disk space) associated to them, whereas a V.V. would not (as the memory would be freed once full/session terminates/etc.).*
2. V.V. is up-to-date, while M.V. is dated based on when created (or last updated).
3. V.V. can be slower than a M.V., depending on the use-case (e.g., time of creation, frequency of view generation, etc.). *Note: M.V. are most often used for data warehousing.*

View Creation Example

Continuing on our 'marvel' example, try the following:

```
CREATE VIEW findall AS SELECT e.cid, e.name, a.name AS alias
FROM extraordinarycitizen e, alias a
WHERE e.cid=a.cid;

SELECT * FROM findall;
```

ISAM

Indexed Sequential Access Method (ISAM), originally developed at IBM for their mainframes, is a file management system. It is used for creating, maintaining, and manipulating files of data so that records (which are composed of fixed length fields, originally stored sequentially in key sequence) can be retrieved sequentially or randomly by one or more keys. Indexes (aka, secondary sets of records) contain pointers to the location of each record, allowing individual records to be retrieved without having to search the entire data set. Indexes of key fields are maintained to achieve fast retrieval of required file records in Indexed files.

B Trees

B Trees are multi-way trees, where each node contains a set of keys and pointers. A B Tree with four keys and five pointers represents the minimum size of a B Tree node. A B Tree contains only data pages.

B Trees are dynamic. That is, the height of the tree grows and contracts as records are added and deleted.

Here is a useful B Tree Visualizer.

B+ Trees

A B+ Tree combines features of ISAM and B Trees. It contains index pages and data pages. The data pages always appear as leaf nodes in the tree. The root node and intermediate nodes are always index pages. These features are similar to ISAM. Unlike ISAM, overflow pages are not used in B+ trees.

The index pages in a B+ tree are constructed through the process of inserting and deleting records. Thus, B+ trees grow and contract like their B Tree counterparts. The contents and the number of index pages reflects this growth and shrinkage.

B+ Trees and B Trees use a "fill factor" to control the growth and the shrinkage. A 50% fill factor would be the minimum for any B+ or B tree. Our example we use the smallest page structure. This means that our B+ tree conforms to the following guidelines:

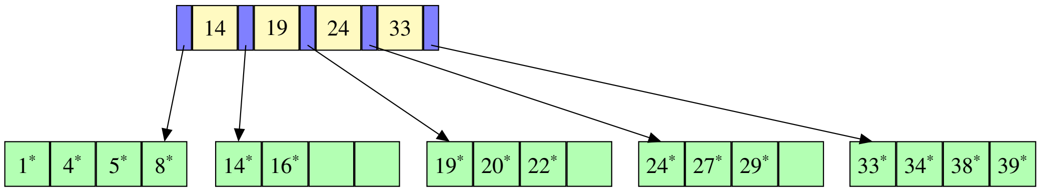
- Number of Keys/page: 4
- Number of Pointers/page: 5
- Fill Factor: 50% (*the root is not bound by this*)
- Minimum Keys in each page: 2 (*the root is not bound by this*)

Here is a useful B+ Tree Visualizer.

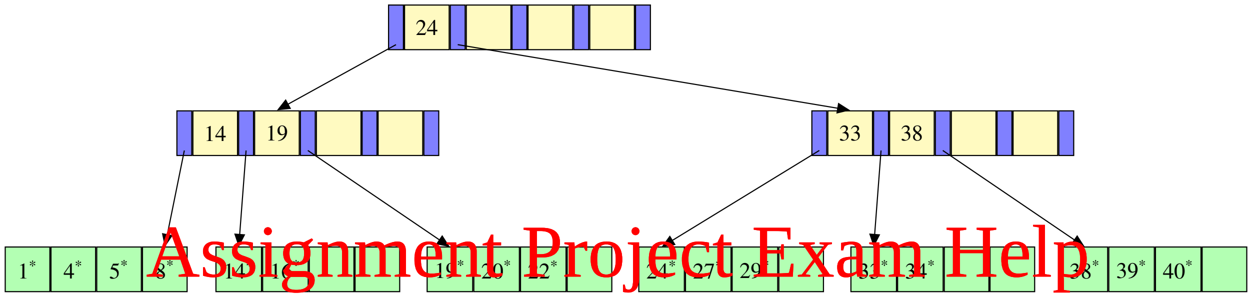
Class Exercise!

Given the following B+ Trees, complete the following action:

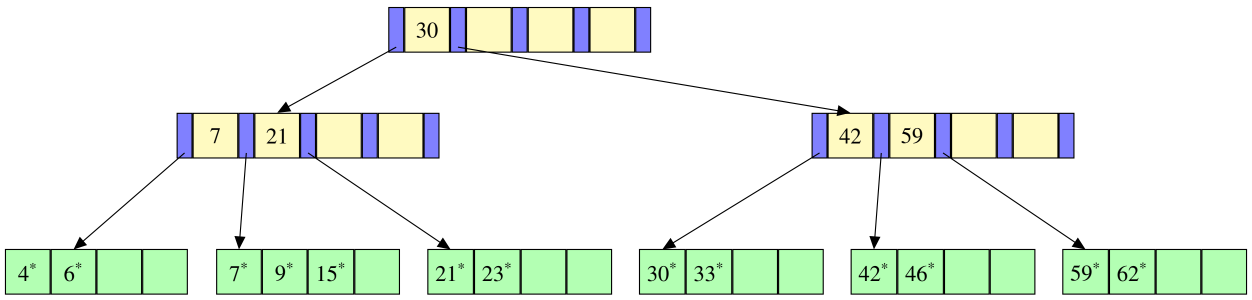
a. Insert 40 into the B+ Tree below.



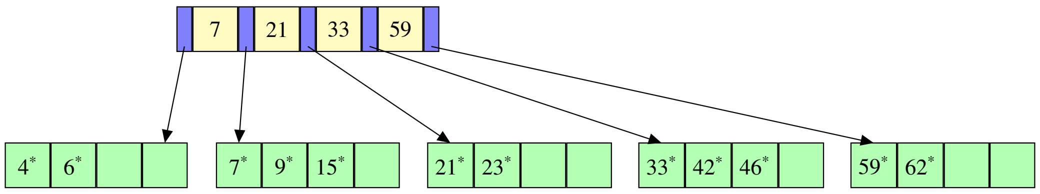
Answer:



b. Delete 30 from the B+ Tree below.



Answer:


7
21
33
59
4*
6*
7*
9*
15*
21*
23*
33*
42*
46*
59*
62*