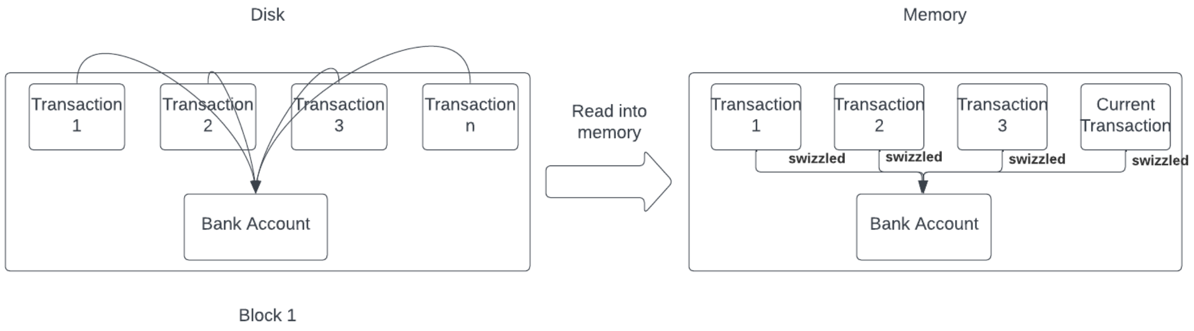
1. Briefly explain for each of the following which is more efficient, completely stating your assumptions. Give a concrete example why by drawing a disk state and a memory state and sequence for record accesses that illustrate your reasoning:

Definition:

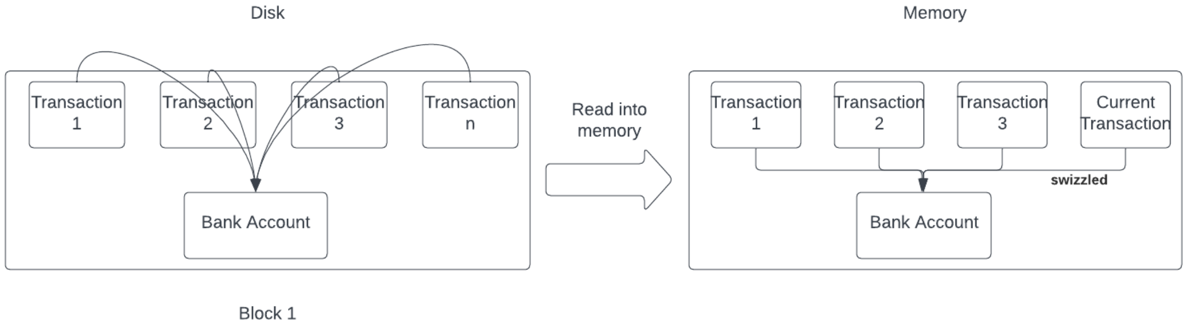
- **Automatic Swizzling -** when read in a block, update the translation table, and swizzle all pointers to this block.
- **Swizzling on Demand -** when we read in a block, update the translation table, and only swizzle a pointer when try to de-reference it.
- **No swizzling -** when read in a block, update the translation table. If try to de-reference a pointer check if in table, if yes, go to where table says, else read in block.
- **Programmer Controlled Swizzling -** sometimes app programmer allowed to force a swizzle.

a) swizzling on demand versus automatic swizzling if you are both logging ATM transactions and updating bank account balances,

Automatic Swizzling



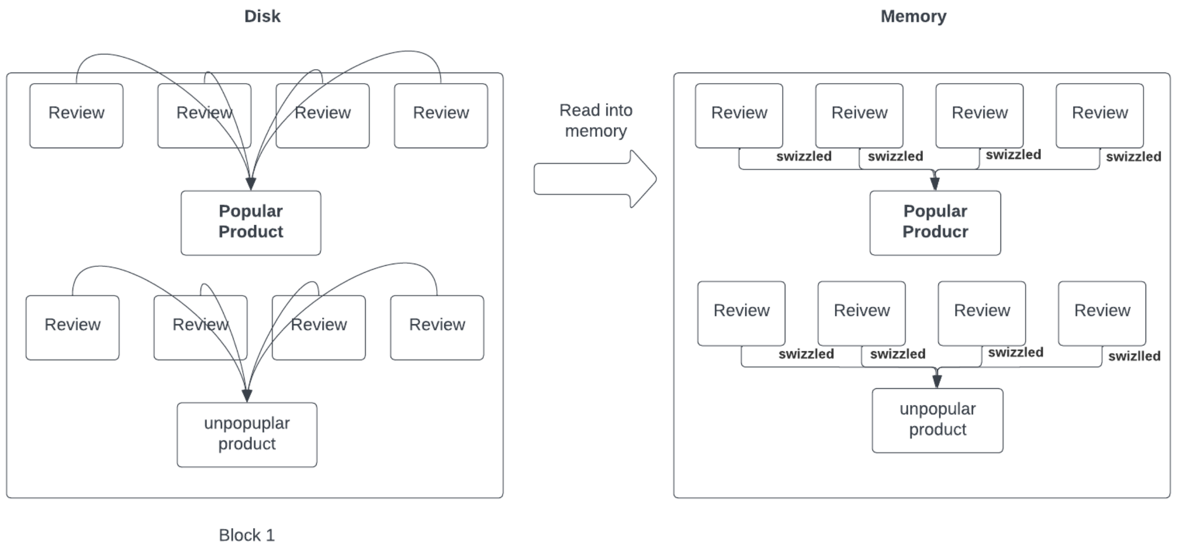
swizzling on demand



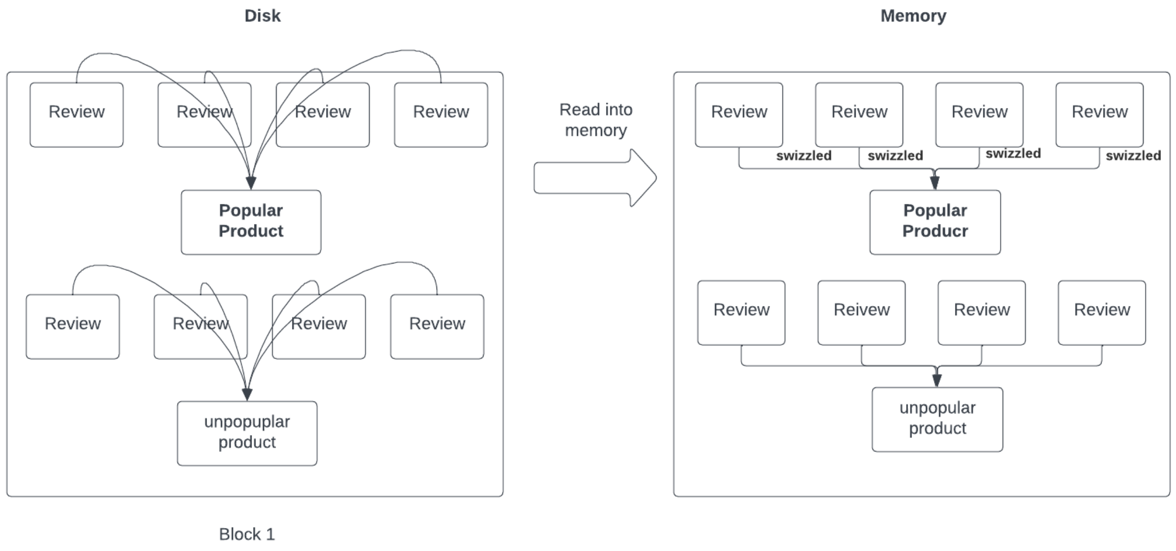
Assume that an account might hold multiple transactions such as past transactions and pending transactions. As a result, there would be many transaction references to an account. For swizzling on demand, when reading the block in memory and attempting to do a transaction, we only dereference a single pointer of the current transaction to the account. When updating the balance, we need to unswizzle a single pointer before writing the updated account record back to disk. In contrast, for automatic swizzling, when reading the block in memory we swizzle all transaction pointers to that account.Indeed, when updating the account, we will have a lot of overhead from unswizzling all pointers before writing back to disk. Thus, swizzling on demand is more efficient than automatic swizzling.

b) swizzling on demand versus automatic swizzling for looking up a popular product and its reviews,

Automatic Swizzling

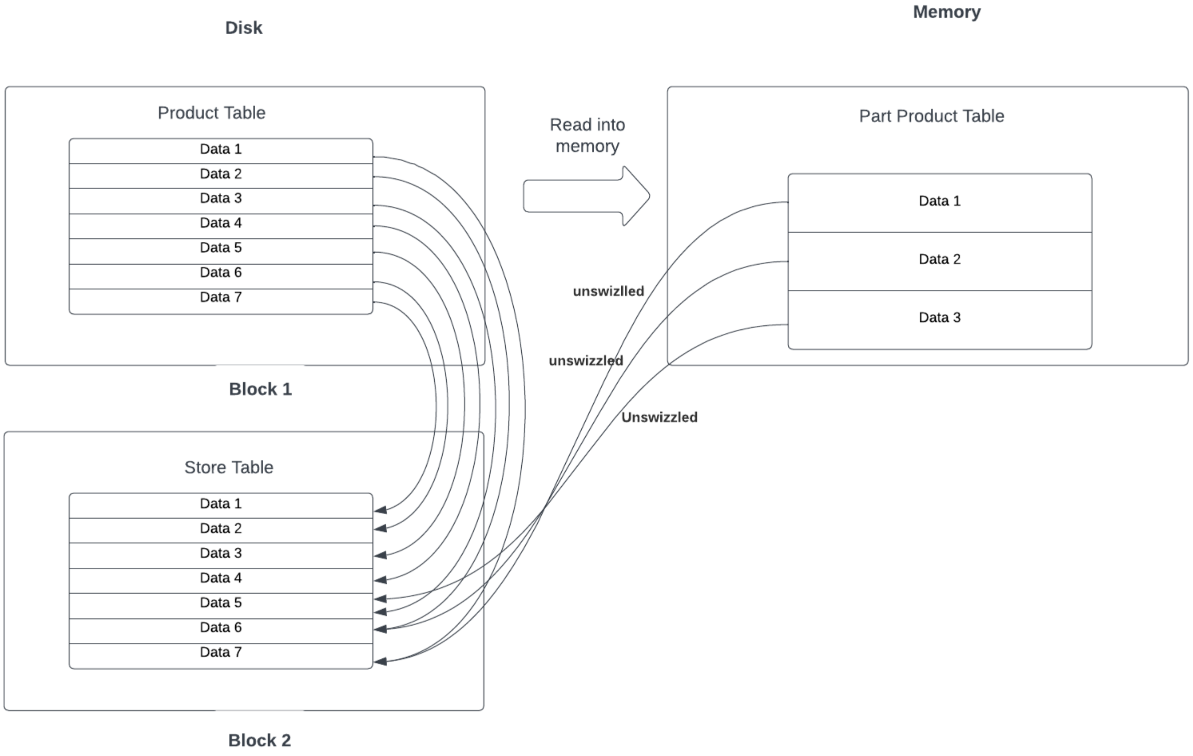


Automatic Swizzling



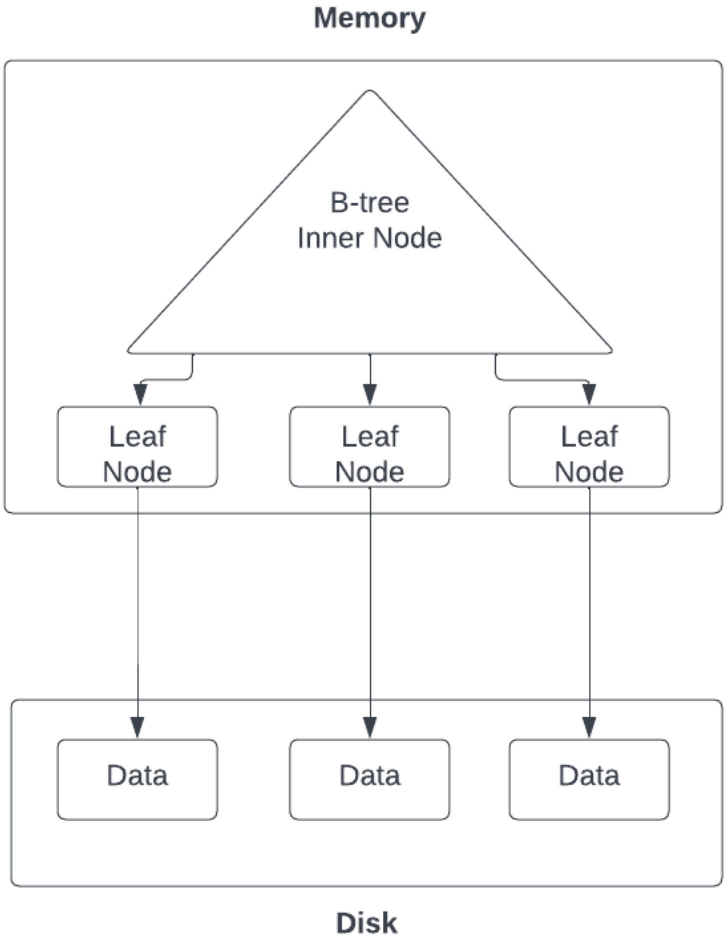
Assume that each review record will have a foreign key reference to a product. For the automatic swizzling when we load the data about products into memory, we will dereference pointers from all reviews to all products including products that we might not need to look up. Therefore, we might spend a lot of time dereferenced many unused pointers. In contrast, for swizzling on demand, we only swizzle the pointer related to the searched product. Thus, swizzling on demand is more efficient than automatic swizzle since we only spend time swizzling pointers related to the product that we look up.

c) no swizzling versus automatic swizzling if you are doing a table scan,



Assume that the data from the table is very large and likely not fit into the memory. Also, assume that record in the table will have reference to some other table. Thus, scanning the table will involve reading the entire table from disk. In other words, we might have to sequentially read the partition of the table from disk to memory. Let's say we scan the table product for the cheapest price item, and the product record has a pointer to the store record. In the case of no swizzling, we just sequentially load the partition of the table into memory and look at the price. However, in the case of automatically swizzling we have to spend time swizzling the pointer to store which we might not need to use. Besides, if the store table is not loading to memory, the pointer will not get dereferenced either while we waste time to check if the record is in memory or not. Thus, no swizzling will be more efficient than automatic swizzling.

d) swizzling on demand versus no swizzling for frequent B-tree looks ups of the same B-tree.



The B-tree pointers will hold the memory address in disk. About the swizzling, the swizzle only dereferences the address in memory. Thus, for swizzling on demand, when looking up and reading a from B-tree, the pointer of B-tree will not get swizzled since the pointer address is to disk. In other words, we might spend some time checking if the block data or address belongs to the memory while failing to change the pointer. In the case of no swizzling, we just directly use the pointer to disk address and read the data from disk. Thus, no swizzling will be much more effective compared to swizzling on demand since we can avoid an extra computation which brings zero benefit to the performance.

2. Suppose a record has the following fields in this order: An 4 byte integer, character string of length 21, two SQL dates, and a 8 bit Binary field. How many bytes does the record take if:

Assumption:
+ SQL date take 2 bytes
+ Each character of String need 1 bytes

a) Fields can start at any byte.

Total bytes
= integer + char_string + 2 * date + binary
= 4 + 21 + 2 * 2 + (8 bits / 8)
= 30 bytes

b) Fields must start at a byte that is a multiple of 4.

Because of fields start at a byte that is a multiple of 4:
+ Date start 3 bytes after char_string field
+ Also assume that we only consider the multiple of 4 when starting but not ending, so it is appropriate for not adding bytes offset at the end of record to make the whole record a multiple of 4.

Total bytes
= integer + char_string + 2 * date + binary
= 4 + 21 + **(3)** + 2 * 2 + 1
= 33 bytes

3. Suppose we are storing 128 byte records into a data file in our database. Each record has a 8 byte integer key. The data file is sorted by this key. Database blocks are 8192 bytes long. Block pointers are 24 bytes. A record pointer consists of a block pointer followed by two byte number indicating which record offset in the block header is for the record in question. Each block begins with two bytes indicating the number of records in a block, followed by a two byte offset pointing to the first element in the available space list, followed by a list of offsets for each record in the block. Suppose we want to store 75 million records. Calculate showing all work the number of blocks used by the data file and by an index on this data file if:

Summary:

Record_size = 128 bytes
Record_key = 8 bytes
Block_size = 8192 bytes
Block_pointer = 24 bytes
Record_pointer = 2 bytes
number_of_record_cost = 2 bytes
Space_pointer = 2 bytes

Assume:

+offset for each record in block is 2 bytes

Note:

Block real capacity is:

Block_capacity = Block_size - number_of_record_cost - Space_pointer
= 8192 - 2 - 2 = 8188 bytes

Number of blocks used by the data file is the same for both case:

Record_cost = Record_size + offset = 128 + 2 = 130 bytes
Record per block = block_capacity / record_size= 8188/130 =62 records / blocks
Total blocks for record = total_records / record_per_block = 75*10^6 / 62 = 1209677 blocks

a) the index is dense,
Dense index have index for each records, so number of index = number of record
Index_size = Record_key + Record_pointer
= Record_key + Block_pointer + 2 bytes_offset = 8 + 24 + 2 = 34 bytes

index_per_block = block_capacity / index_size = 8188 / 34 = 238 index/block

Total block for dense index= total_index / index_per_block = 75*10^6 / 238 = 315126 blocks

b)the index is sparse.

Sparse index have index for each block of record, so number of index = number of block for record, which is 1209677 blocks

Index_size = Record_key + Block_pointer = 8 + 24 = 32 bytes

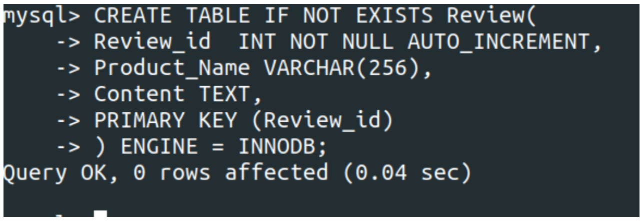
index_per_block = block_capacity / index_size = 8188 / 32 = 255 index/block

Total block for spare index= total_index / index_per_block = 1209677 / 255 = 4744 blocks

4. Create a text transcript of doing the following activities on a Mysql database from the Mysql shell: Create a Mysql table suitable for handling online shopping product reviews. Add a full text index on the columns you use for the text contents of your post. Insert a few blog posts. Determine what the current stop words are for your index. Do a search of your table on a word or phrase that is not a stop word, but is in one of the inserted posts. Do a search of your table on a word that is not a stop word, and is in one of the inserted posts. Change the stop list to now include your non-stop word and redo the searches. Please consult Mysql Full Text Stopwords page, for more info about how to do this problem.

Create Table:

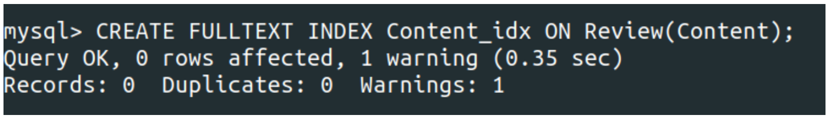
```
CREATE TABLE IF NOT EXISTS Review(
Review_id  INT NOT NULL AUTO_INCREMENT,
Product_Name VARCHAR(256),
Content TEXT,
PRIMARY KEY (Review_id)
) ENGINE = INNODB;
```



```
mysql> CREATE TABLE IF NOT EXISTS Review(
    -> Review_id  INT NOT NULL AUTO_INCREMENT,
    -> Product_Name VARCHAR(256),
    -> Content TEXT,
    -> PRIMARY KEY (Review_id)
    -> ) ENGINE = INNODB;
Query OK, 0 rows affected (0.04 sec)
```

Create Index:

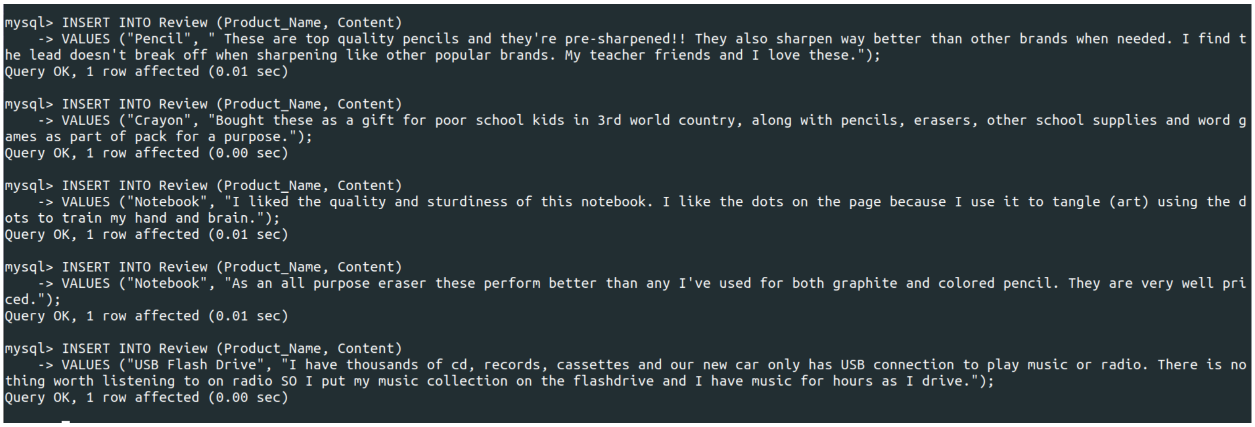
```
CREATE FULLTEXT INDEX Content_idx ON Review(Content);
```



```
mysql> CREATE FULLTEXT INDEX Content_idx ON Review(Content);
Query OK, 0 rows affected, 1 warning (0.35 sec)
Records: 0  Duplicates: 0  Warnings: 1
```

Insert Data:

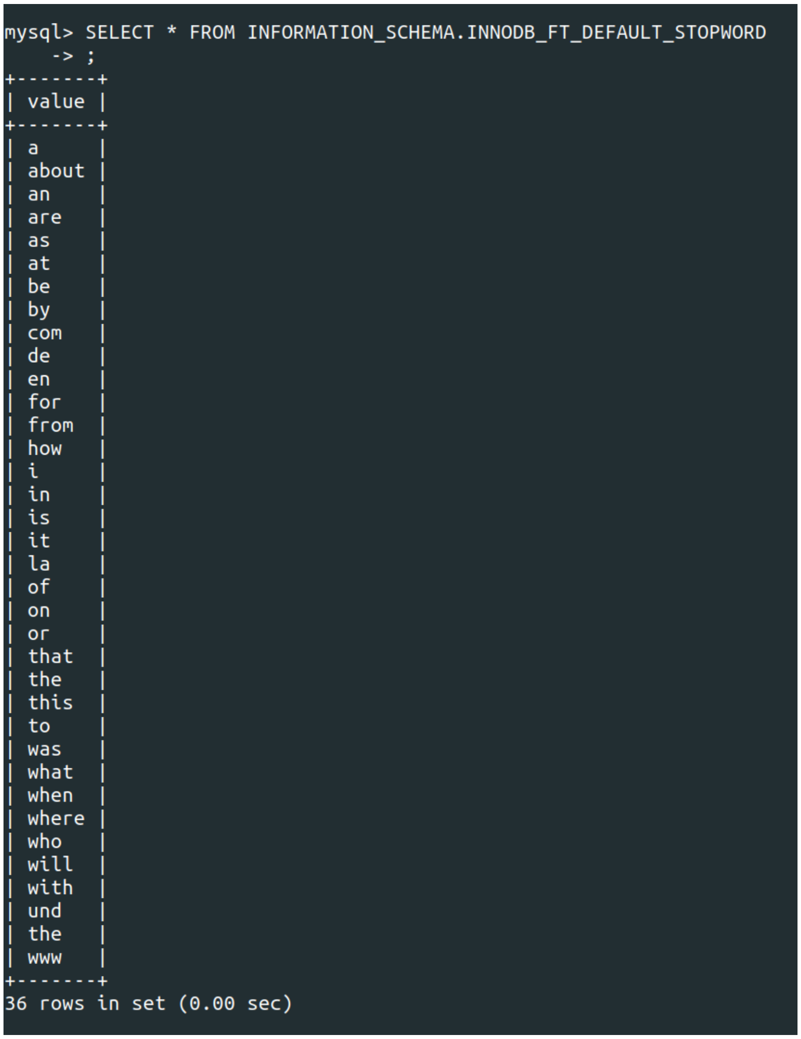
```
INSERT INTO Review (Product_Name, Content)
VALUES ("Pencil", " These are top quality pencils and they're pre-sharpened!! They also sharpen way better than other brands when needed. I find the lead doesn't break off when sharpening like other popular brands. My teacher friends and I love these.");

INSERT INTO Review (Product_Name, Content)
VALUES ("Crayon", "Bought these as a gift for poor school kids in 3rd world country, along with pencils, erasers, other school supplies and word games as part of pack for a purpose.");

INSERT INTO Review (Product_Name, Content)
VALUES ("Notebook", "I liked the quality and sturdiness of this notebook. I like the dots on the page because I use it to tangle (art) using the dots to train my hand and brain.");

INSERT INTO Review (Product_Name, Content)
VALUES ("Notebook", "As an all purpose eraser these perform better than any I've used for both graphite and colored pencil. They are very well priced.");

INSERT INTO Review (Product_Name, Content)
VALUES ("USB Flash Drive", "I have thousands of cd, records, cassettes and our new car only
has USB connection to play music or radio. There is nothing worth listening to on radio SO I put
my music collection on the flashdrive and I have music for hours as I drive.");
```



```
mysql> INSERT INTO Review (Product_Name, Content)
    -> VALUES ("Pencil", " These are top quality pencils and they're pre-sharpened!! They also sharpen way better than other brands when needed. I find t
he lead doesn't break off when sharpening like other popular brands. My teacher friends and I love these.");
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO Review (Product_Name, Content)
    -> VALUES ("Crayon", "Bought these as a gift for poor school kids in 3rd world country, along with pencils, erasers, other school supplies and word g
ames as part of pack for a purpose.");
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Review (Product_Name, Content)
    -> VALUES ("Notebook", "I liked the quality and sturdiness of this notebook. I like the dots on the page because I use it to tangle (art) using the d
ots to train my hand and brain.");
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO Review (Product_Name, Content)
    -> VALUES ("Notebook", "As an all purpose eraser these perform better than any I've used for both graphite and colored pencil. They are very well pri
ced.");
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO Review (Product_Name, Content)
    -> VALUES ("USB Flash Drive", "I have thousands of cd, records, cassettes and our new car only has USB connection to play music or radio. There is no
thing worth listening to on radio SO I put my music collection on the flashdrive and I have music for hours as I drive.");
Query OK, 1 row affected (0.00 sec)
```

Check Default Stop Word:

```
SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_DEFAULT_STOPWORD;
```



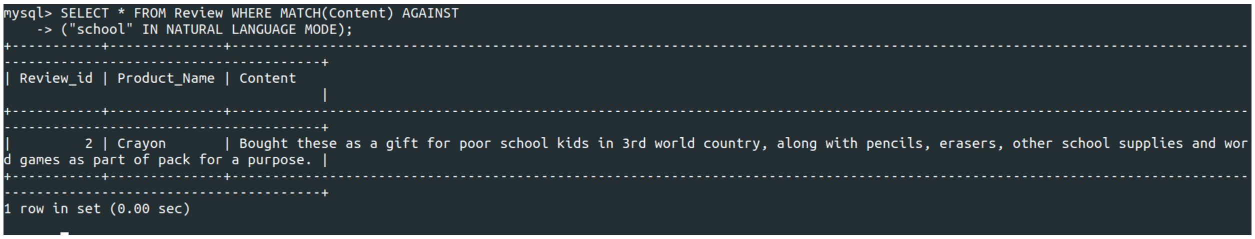
```
mysql> SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_DEFAULT_STOPWORD
    -> ;
+-------+
| value |
+-------+
| a     |
| about |
| an    |
| are   |
| as    |
| at    |
| be    |
| by    |
| com   |
| de    |
| en    |
| for   |
| from  |
| how   |
| i     |
| in    |
| is    |
| it    |
| la    |
| of    |
| on    |
| or    |
| that  |
| the   |
| this  |
| to    |
| was   |
| what  |
| when  |
| where |
| who   |
| will  |
| with  |
| und   |
| the   |
| www   |
+-------+
36 rows in set (0.00 sec)
```

From table above stop word in my text content is :

A, an, are, as, for, i, in, is, it, of, on, or, the, this, to, when, with

A search of my table on a word or phrase that is not a stop word

SELECT * FROM Review WHERE MATCH(Content) AGAINST
("school" IN NATURAL LANGUAGE MODE);

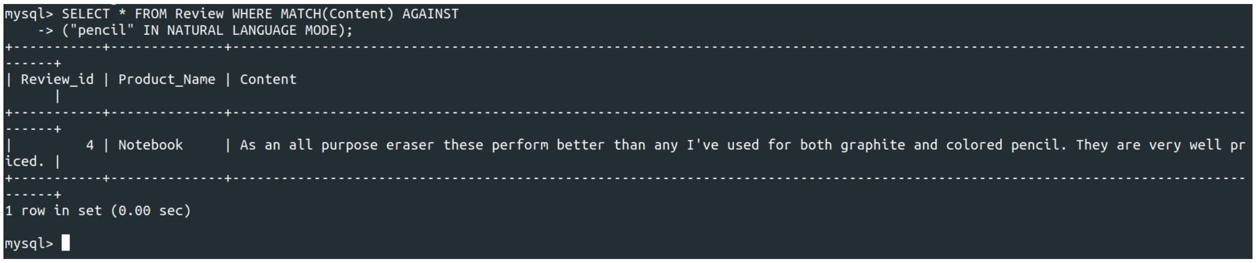


```
mysql> SELECT * FROM Review WHERE MATCH(Content) AGAINST
    -> ("school" IN NATURAL LANGUAGE MODE);
+-----------+--------------+-----------------------------------------------------------------------------------------------------------------------------
---------------------------------------+
| Review_id | Product_Name | Content
                                       |
+-----------+--------------+-----------------------------------------------------------------------------------------------------------------------------
---------------------------------------+
|         2 | Crayon       | Bought these as a gift for poor school kids in 3rd world country, along with pencils, erasers, other school supplies and wor
d games as part of pack for a purpose. |
+-----------+--------------+-----------------------------------------------------------------------------------------------------------------------------
---------------------------------------+
1 row in set (0.00 sec)
```

A search of your table on a word that is not a stop word, then add it to to stop word list and search again:

Search for “pencil”

SELECT * FROM Review WHERE MATCH(Content) AGAINST
("pencil" IN NATURAL LANGUAGE MODE);



```
mysql> SELECT * FROM Review WHERE MATCH(Content) AGAINST
    -> ("pencil" IN NATURAL LANGUAGE MODE);
+-----------+--------------+-----------------------------------------------------------------------------------------------------------------------------
------+
| Review_id | Product_Name | Content
      |
+-----------+--------------+-----------------------------------------------------------------------------------------------------------------------------
------+
|         4 | Notebook     | As an all purpose eraser these perform better than any I've used for both graphite and colored pencil. They are very well pr
iced. |
+-----------+--------------+-----------------------------------------------------------------------------------------------------------------------------
------+
1 row in set (0.00 sec)

mysql> 
```

The search return one records

Add “pencil” to stop word:

# Prepare new stop word list
CREATE TABLE my_stopwords(value VARCHAR(30)) ENGINE = INNODB;

# Add stopword from my content which also exist in DEFAULT_STOPWORD to list
INSERT INTO my_stopwords(value) VALUES("a");
INSERT INTO my_stopwords(value) VALUES("an");
INSERT INTO my_stopwords(value) VALUES("are");
INSERT INTO my_stopwords(value) VALUES("as");
INSERT INTO my_stopwords(value) VALUES("for");
INSERT INTO my_stopwords(value) VALUES("i");
INSERT INTO my_stopwords(value) VALUES("in");
INSERT INTO my_stopwords(value) VALUES("is");
INSERT INTO my_stopwords(value) VALUES("it");
INSERT INTO my_stopwords(value) VALUES("of");
INSERT INTO my_stopwords(value) VALUES("on");
INSERT INTO my_stopwords(value) VALUES("or");
INSERT INTO my_stopwords(value) VALUES("the");
INSERT INTO my_stopwords(value) VALUES("this");
INSERT INTO my_stopwords(value) VALUES("to");
INSERT INTO my_stopwords(value) VALUES("when");
INSERT INTO my_stopwords(value) VALUES("with");

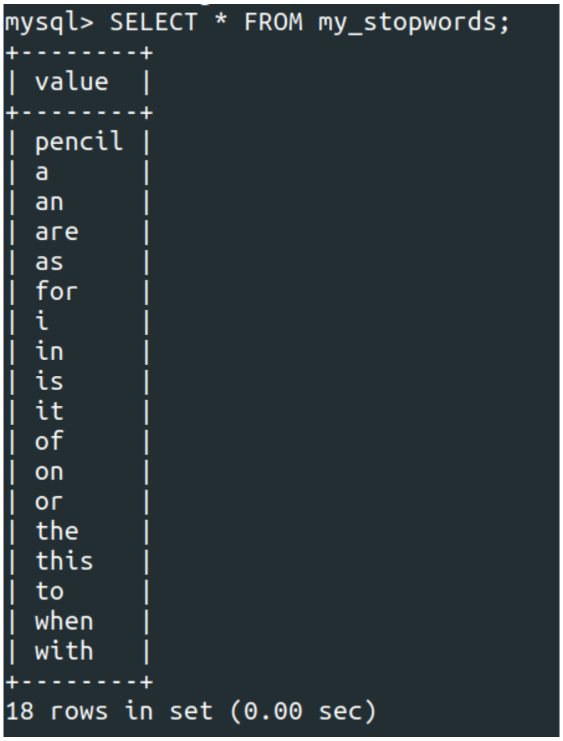
# Add pencil to that list as well
INSERT INTO my_stopwords(value) VALUES("pencil");

# Set Stopword list to list “my_stopwords”
SET GLOBAL innodb_ft_server_stopword_table = 'hw2/my_stopwords';
SET GLOBAL innodb_ft_aux_table='hw2/Review';

#Drop old index:
ALTER TABLE Review
DROP INDEX Content_idx;

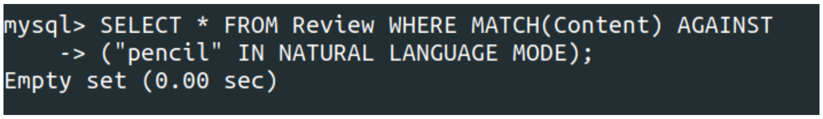
#Create index that use new stopword list:
CREATE FULLTEXT INDEX Content_idx ON Review(Content);

# New stopwords:
SELECT * FROM my_stopwords;



```
mysql> SELECT * FROM my_stopwords;
+--------+
| value  |
+--------+
| pencil |
| a      |
| an     |
| are    |
| as     |
| for    |
| i      |
| in     |
| is     |
| it     |
| of     |
| on     |
| or     |
| the    |
| this   |
| to     |
| when   |
| with   |
+--------+
18 rows in set (0.00 sec)
```

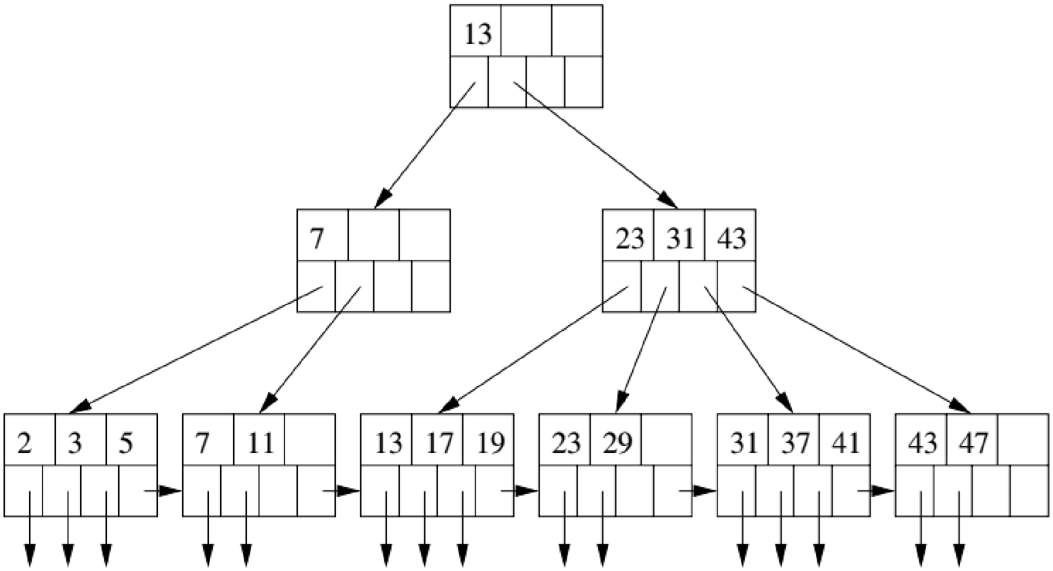
Search for “pencil” again:



```
mysql> SELECT * FROM Review WHERE MATCH(Content) AGAINST
    -> ("pencil" IN NATURAL LANGUAGE MODE);
Empty set (0.00 sec)
```

Now we cannot search for “pencil”

5. For the following B-tree, show:



a) step-by-step the blocks accessed in looking up 15 (a key not in the tree)

From root, 15 > 13, so we follow second pointer to node [23, 31, 43]

15 < 23, so we follow the first pointer to node [13, 17, 19]

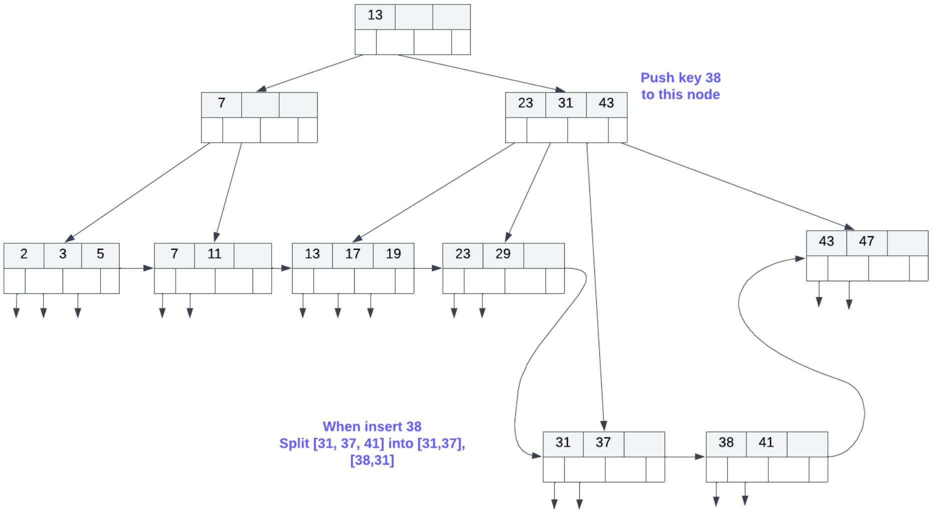
Node [13, 17, 19] is leaf node, but there is no block that has 15. Thus we conclude there is no record for 15.

b) step-by-step inserting a new entry for a record with key 38

From root 38 > 13, so we follow second pointer to node [23, 31, 43]

Then, 31 <= 38 < 41, we follow third pointer to node [31, 37, 41]

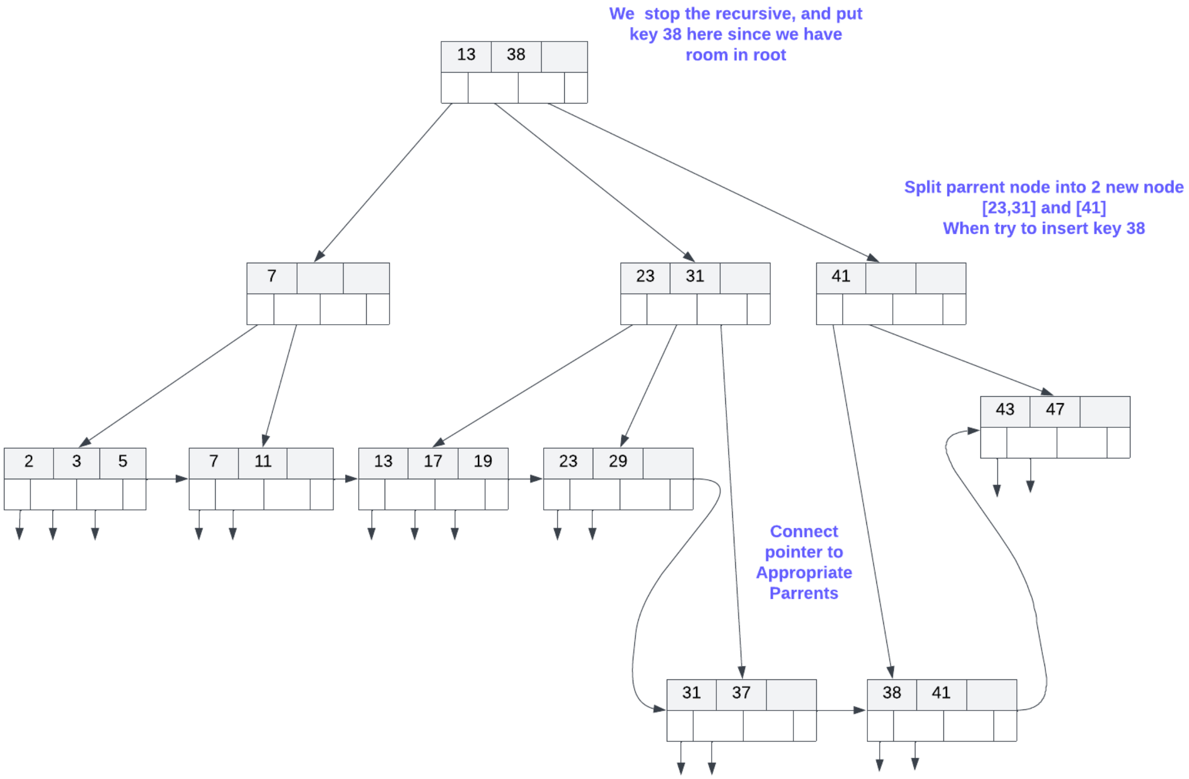
This is the leaf node so we insert 38 to this node. However, the node is full so we split them into 2 node [31, 37] and [38, 41], and push key 38 to parent node as the figure below



Push key 38 to this node

When insert 38
Split [31, 37, 41] into [31,37], [38,31]

Because parent is full, we recursively split parent into 2 nodes [23, 31] and [41]. Then, we connect a pointer from two parents to their children nodes. Finally, we have to push key 38 to the root node [13]. Since root node still have room for key 38, we insert 38 there as the figure below:



We stop the recursive, and put key 38 here since we have room in root

Split parrent node into 2 new node [23,31] and [41]
When try to insert key 38

Connect pointer to Appropriate Parrents