Bulk Loading, MySQL Page Internals and Fragmentation

Bulk loading Rules

- Creation of B+ Tree one by one is expensive

1. Sort the data entries according to the order
2. Allocate an empty page to serve as the root and insert a pointer to the first page of entries into it.
3. When the root is full, split the root and create a new root page.
4. Keep inserting entries to the right most index (node) just above the leaf level until all entries are indexed.

Bulk Loading Example:

a) 3*- 4*

b) 6*- 9*

c) 10* - 11*

d) 12* - 13*

e) 20*- 22*

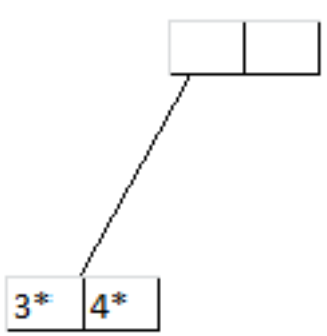
f) 23* - 31*

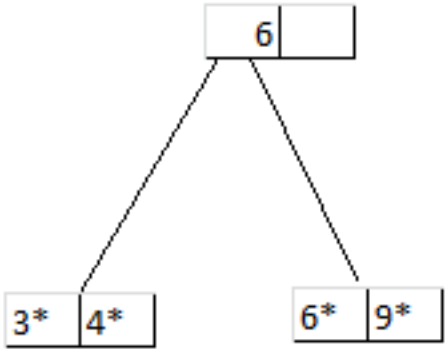
g) 35* - 36*

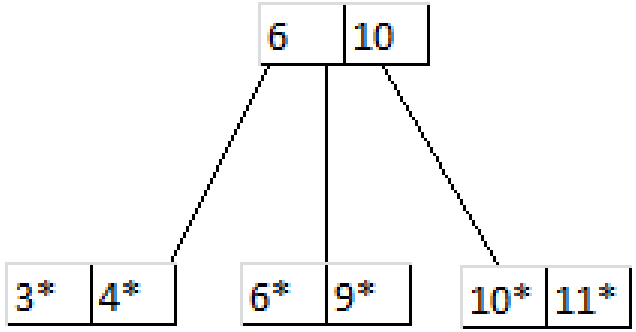
h) 38* - 41*

i) 44* -

Insert a, b, c

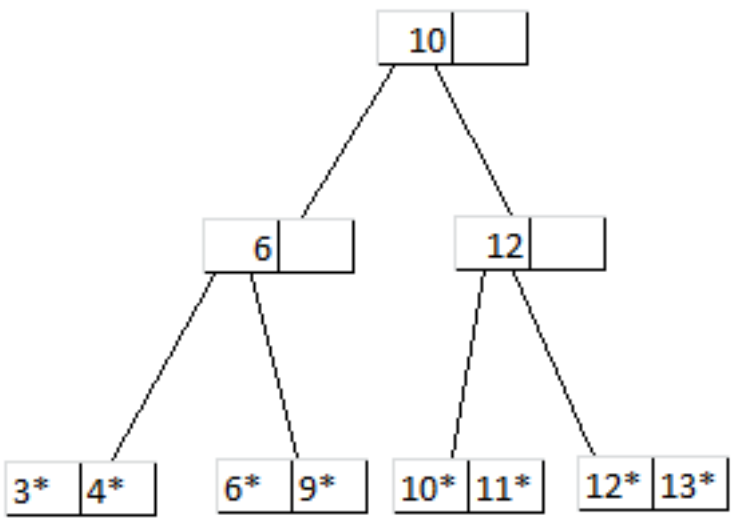


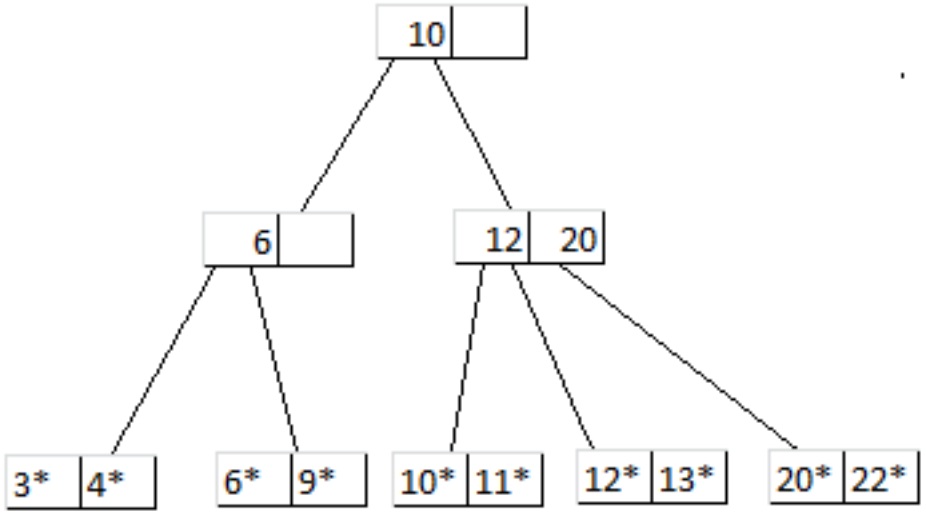




a) 3*- 4* b) 6*- 9* c) 10* - 11* d) 12* - 13* e) 20*- 22* f) 23* - 31* g) 35* - 36* h) 38* - 41* i) 44* -

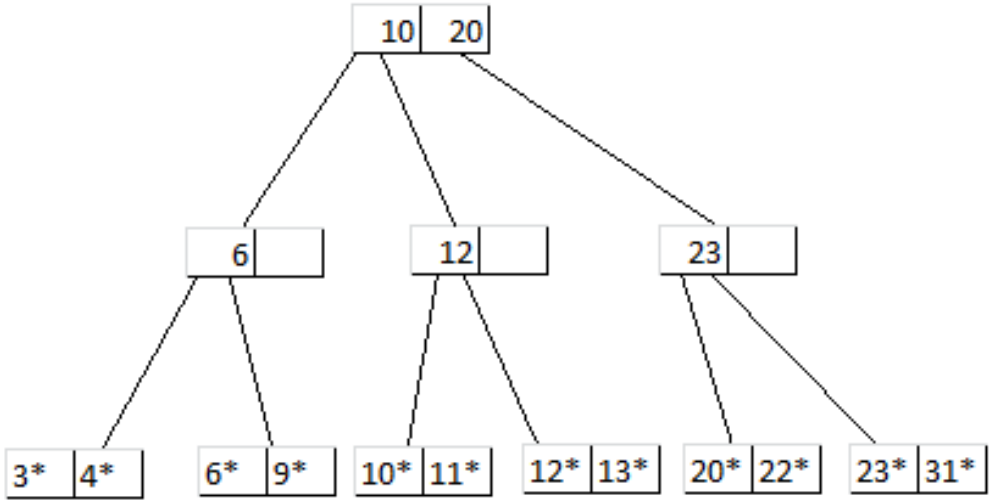
Insert d, e

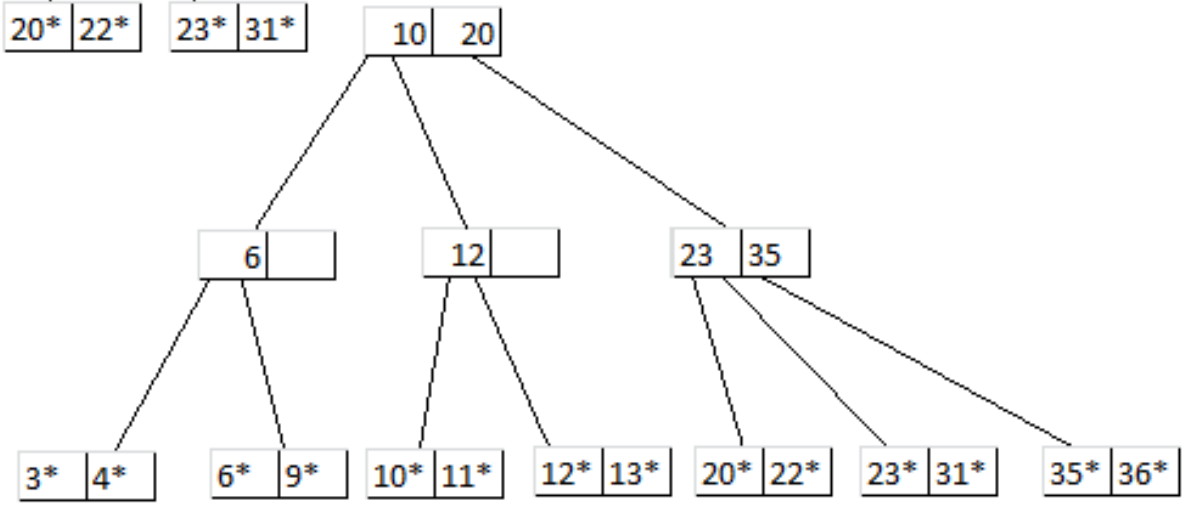




a) 3*- 4* b) 6*- 9* c) 10* - 11* d) 12* - 13* e) 20*- 22* f) 23* - 31* g) 35* - 36* h) 38* - 41* i) 44* -

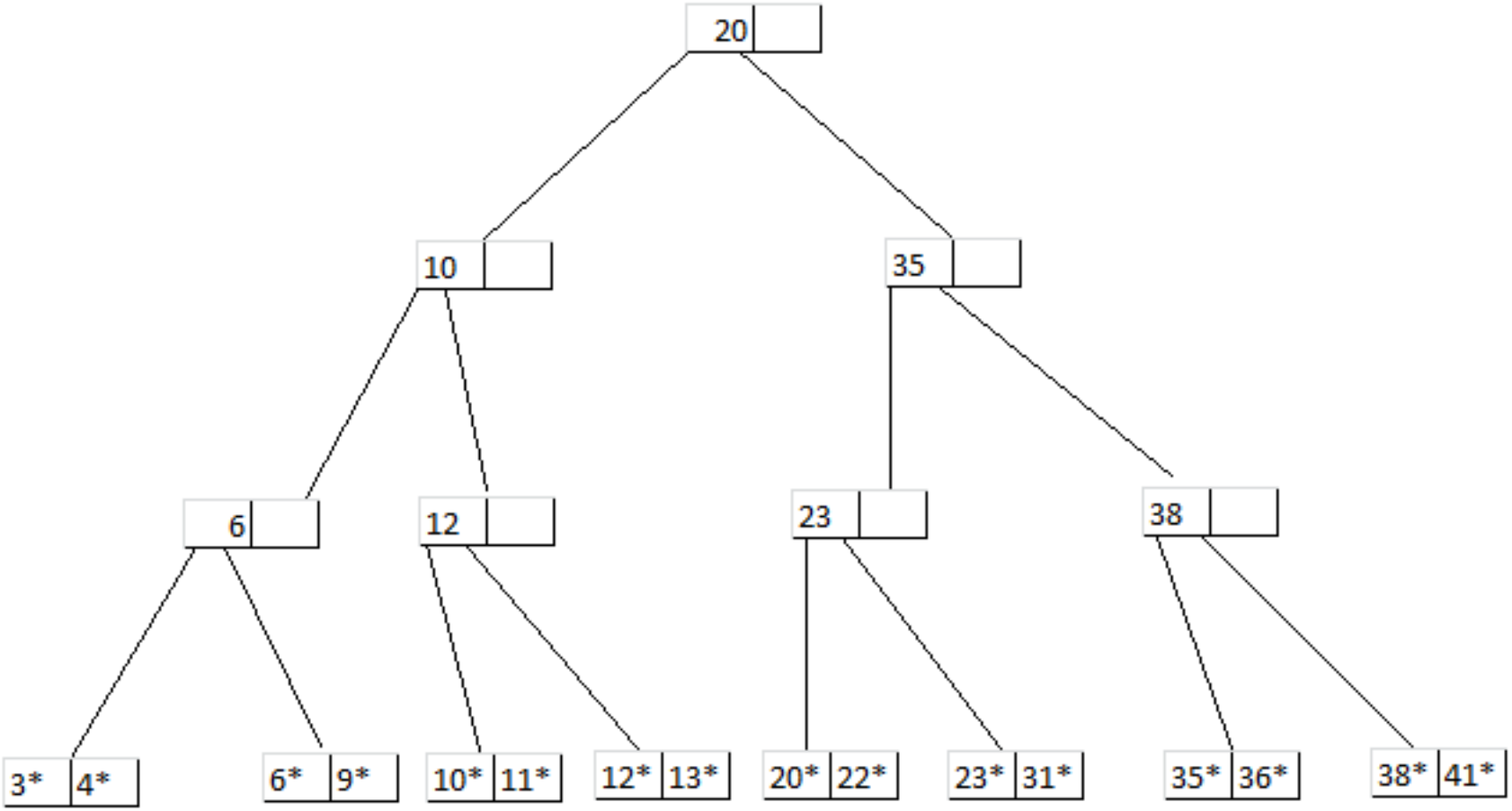
Insert f, g





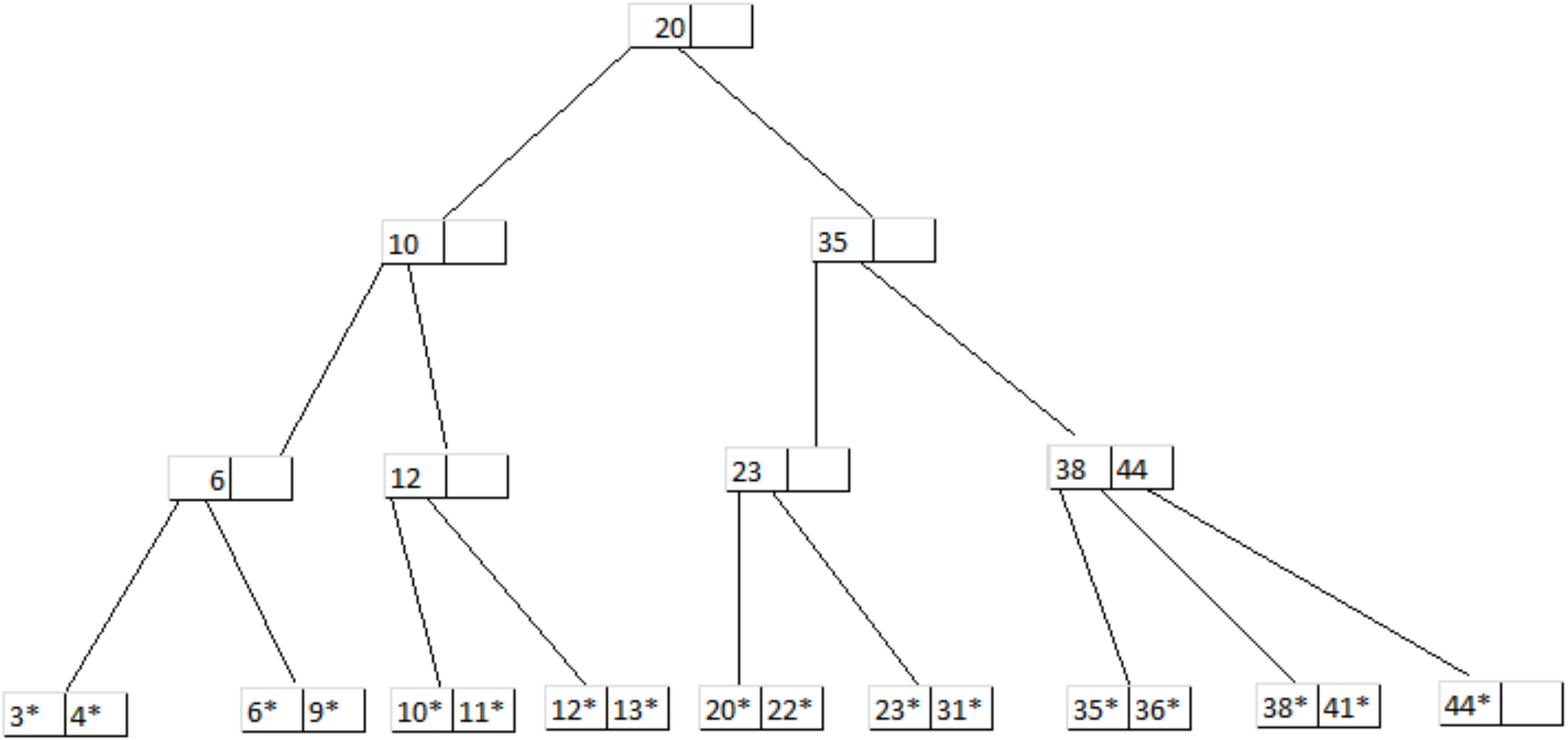
a) 3*- 4* b) 6*- 9* c) 10* - 11* d) 12* - 13* e) 20*- 22* f) 23* - 31*
g) 35* - 36* h) 38* - 41* i) 44* -

Insert h



a) 3*- 4* b) 6*- 9* c) 10* - 11* d) 12* - 13* e) 20*- 22* f) 23* - 31*
g) 35* - 36* h) 38* - 41* i) 44* -

Insert i



a) 3*- 4* b) 6*- 9* c) 10* - 11* d) 12* - 13* e) 20*- 22* f) 23* - 31*
g) 35* - 36* h) 38* - 41* i) 44* -

Bulk loading MySQL

- The bulk load feature is disabled by default for MySQL 8.0 and higher versions.

- In MySQL v8 or higher, add the following lines to my.cnf file and restart the server.
  [mysqld]
  local_infile=ON

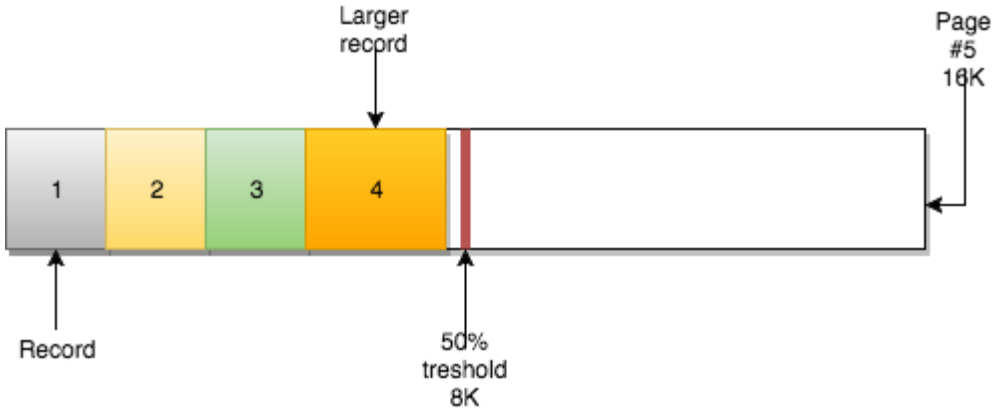
- If the my.cnf file does not exist, create one in the main directory of the server.

Bulk loading MySQL

- show variables like 'max_allowed_packet';
- SET GLOBAL max_allowed_packet=524288000;
- LOAD DATA LOCAL INFILE '/path/to/products.csv' INTO TABLE products;

Page Internals MySQL

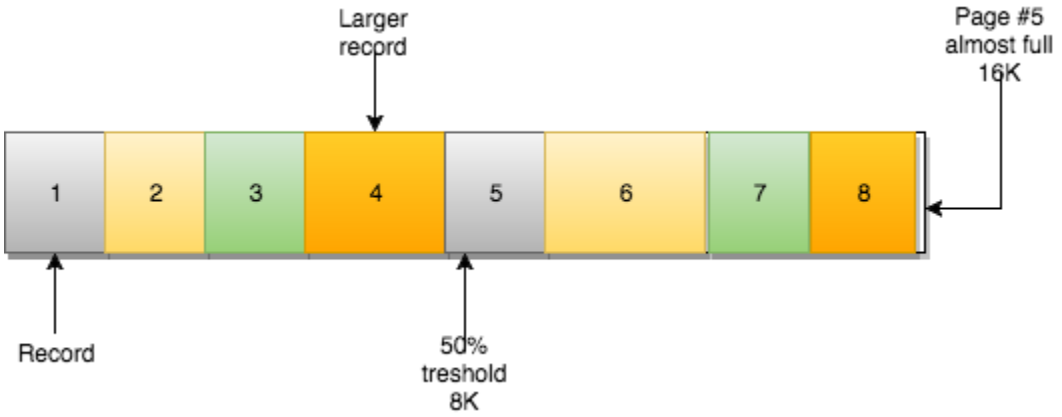
- A page can be empty or fully filled (100%).
- The row-records will be organized by PK.
- If table is using an *AUTO_INCREMENT*, you will have the sequence ID = 1, 2, 3, 4, etc.



https://www.percona.com/blog/2017/04/10/innodb-page-merging-and-page-splitting/

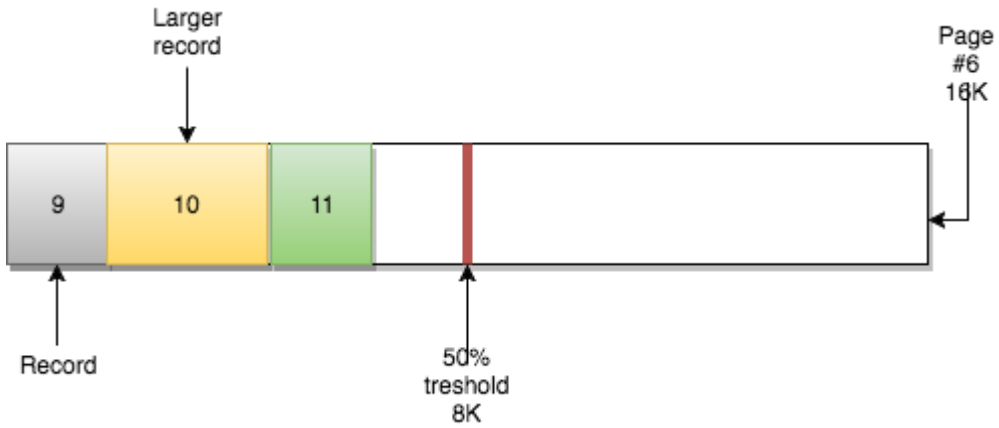
Page Internals MySQL

- A page also has another important attribute: *MERGE_THRESHOLD*.
- The default value of this parameter is 50% of the page, and it plays a very important role in InnoDB merge activity:



Page Internals MySQL

- While inserting data, the page is filled up sequentially if the incoming record can be accommodated inside the page.
- When a page is full, the next record will be inserted into the NEXT page:

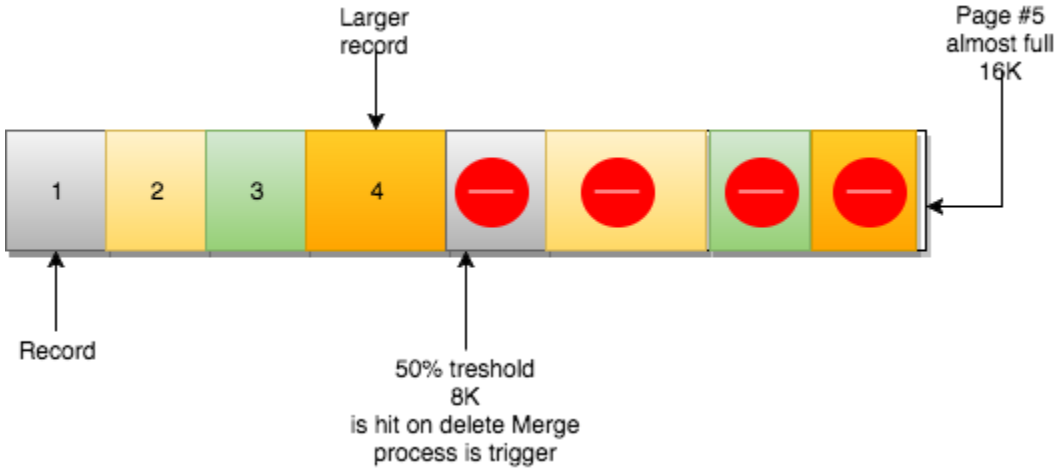


Page Internals MySQL

- Page #5 has a reference to the next page, Page #6.
- Page #6 has references backward to the previous page (Page #5) and a forward to the next page (Page #7)
- This mechanism of a linked list allows for fast, in-order scans (i.e., Range Scans).

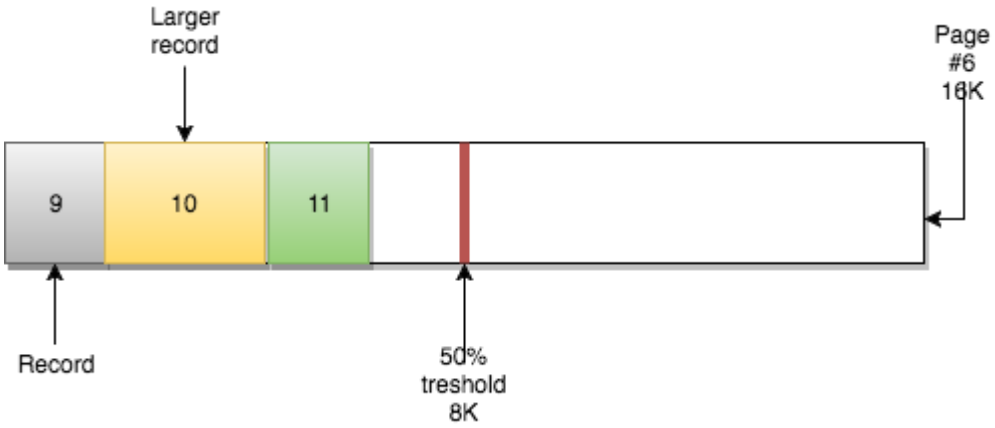
Page Merging MySQL

- On deleting a record, it is not physically deleted, instead, it flags the record as deleted, the space becomes reclaimable.
- When a page has received enough deletes to match the MERGE_THRESHOLD (50% of the page size by default)
- InnoDB starts to look to the closest pages (NEXT and PREVIOUS) to see if merging of two pages is possible



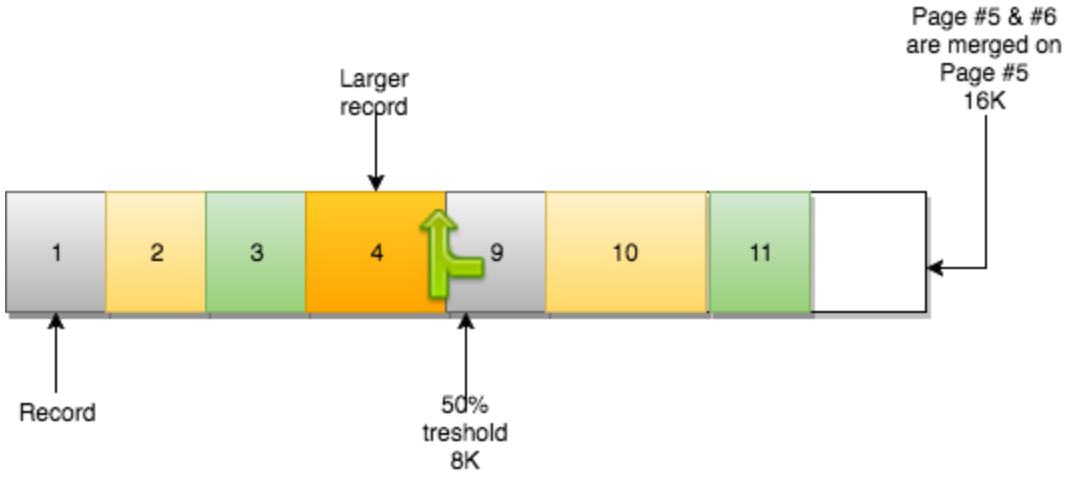
Page Merging MySQL

- MERGE_THRESHOLD is configurable for table and specific indexes.
- In this example, Page #6 is utilizing less than half of its space.
- Page #5 received many deletes and is also now less than 50% used.



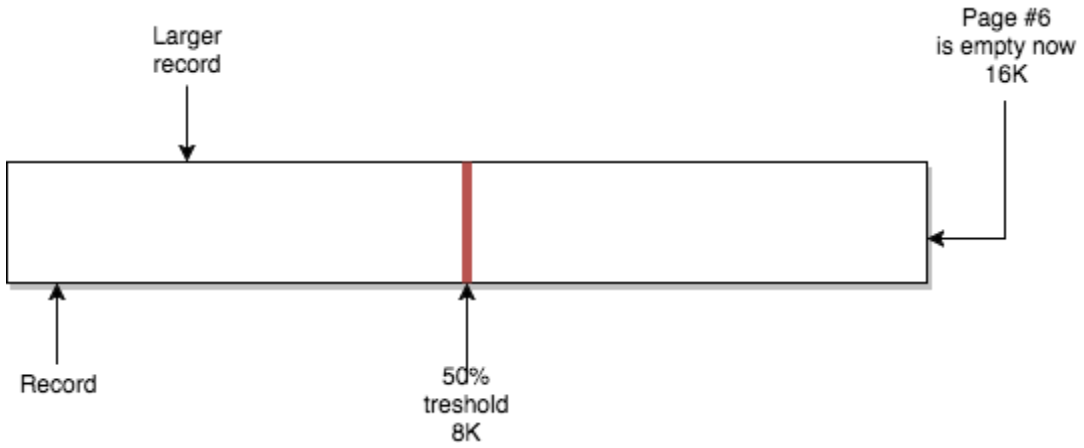
Page Merging MySQL

- From InnoDB's perspective, they are mergeable
- The rule is: *Merges happen on delete and update operations involving close linked pages.*



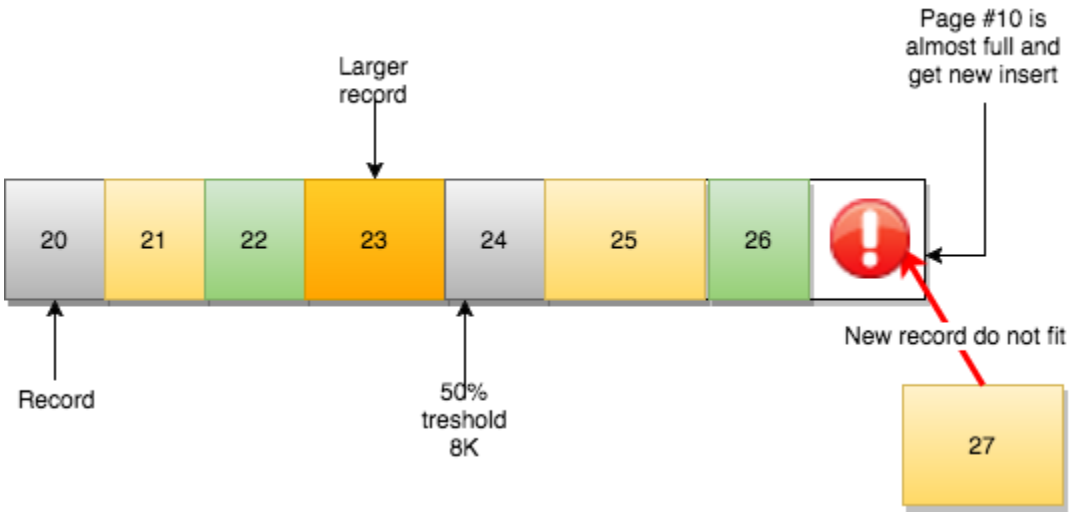
Page Merging MySQL

- The merge operation results in Page #5 containing its previous data plus the data from Page #6.
- Page #6 becomes an empty page, usable for new data.
- On merge operation successful, the *index_page_merge_successful* metric in *INFORMATION_SCHEMA.INNODB_METRICS* incremented.



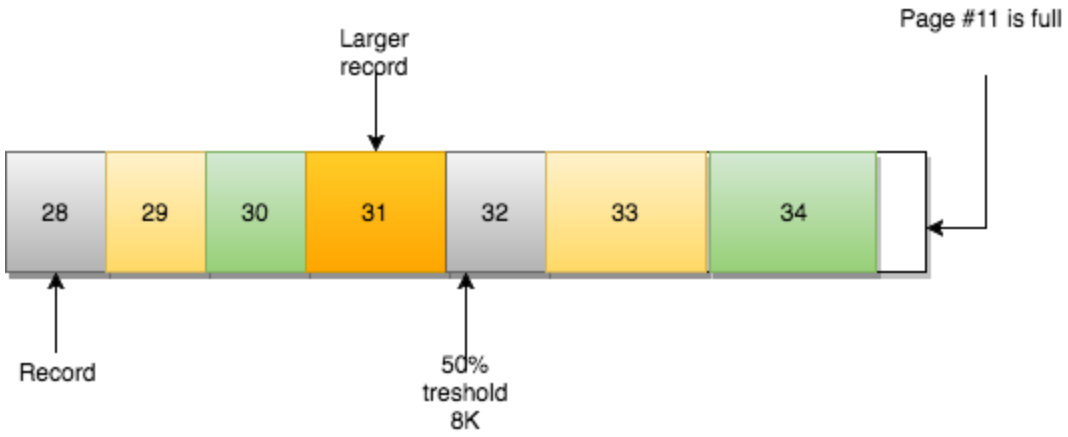
Page Splitting MySQL

- Page #10 doesn't have enough space to accommodate the new (or updated) record.
- Following the next page logic, the record should go on Page #11.



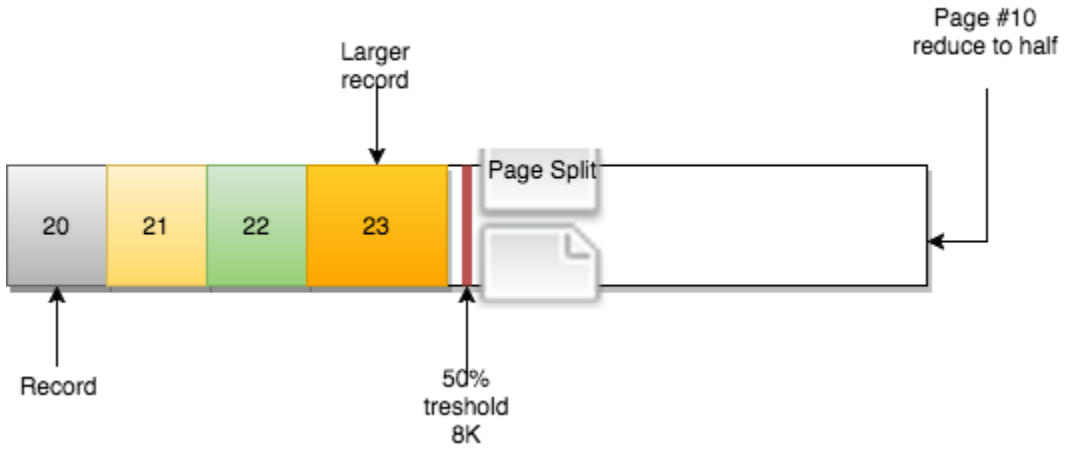
Page Splitting MySQL

- Page #11 is also full, and data cannot be inserted out of order. So what can be done?
- Remember the linked list we spoke about? At this moment Page #10 has Prev=9 and Next=11.



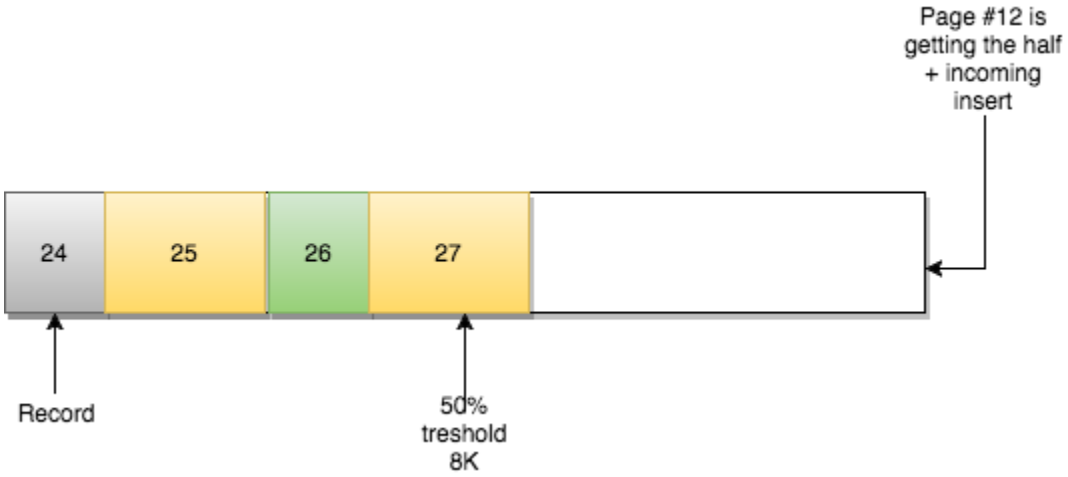
Page Splitting MySQL

- What InnoDB will do is (simplifying):
- Create a new page
- Identify where the original page (Page #10) can be split (at the record level)
- Move records
- Redefine the page relationships



Page Splitting MySQL

- A new Page #12 is created:
- Page #11 stays as it is.
- Page #10 will have Prev=9 and Next=12
- Page #12 Prev=10 and Next=11
- Page #11 Prev=12 and Next=13



Page Splitting MySQL

- Physically the page is located out of order
- Reorganize the data by OPTIMIZE the table.
- This can be a very heavy and long process, but often is the only way to recover from a situation where too many pages are located sparsely
- InnoDB tracks the number of page splits in ***INFORMATION_SCHEMA.INNODB_METRICS***.
- Check *index_page_splits* and *index_page_reorg_attempts/successful* metrics.

MySQL Table Fragmentation

- Table fragmentation with DELETE statements
- Whenever a huge delete operation, in most cases, always rebuilding of table to reclaim the disk space.
- Table fragmentation is happening with the INSERT statements too
- There are three major cases of table fragmentation with INSERTs :
  - INSERT with ROLLBACK
  - Failed INSERT statement
  - Fragmentation with page-splits

https://www.percona.com/blog/2020/06/24/mysql-table-fragmentation-beware-of-bulk-insert-with-failure-or-rollback/

MySQL Table Fragmentation

- Test environment to experiment with those cases.
- DB: percona
- Tables : frag, ins_frag, frag_page_spl
- Table Size: 2G

MySQL Table Fragmentation

Case 1: INSERT with ROLLBACK

```
mysql> create table ins_frag like frag;
Query OK, 0 rows affected (0.01 sec)

mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into ins_frag select * from frag;
Query OK, 47521280 rows affected (3 min 7.45 sec)
Records: 47521280  Duplicates: 0  Warnings: 0

#Linux shell
sakthi-3.2# ls -lrth
total 8261632
-rw-r-----  1 _mysql  _mysql   2.0G Jun 17 02:43 frag.ibd
-rw-r-----  1 _mysql  _mysql   2.0G Jun 17 03:00 ins_frag.ibd
```

MySQL Table Fragmentation

```
mysql> select count(*) from ins_frag;
+----------+
| count(*) |
+----------+
| 47521280 |
+----------+
1 row in set (1.87 sec)

mysql> rollback;
Query OK, 0 rows affected (5 min 45.21 sec)
```

ROLLBACK the INSERT

```
mysql> select count(*) from ins_frag;
+----------+
| count(*) |
+----------+
|        0 |
+----------+
1 row in set (0.00 sec)
```

```
#Linux shell
sakthi-3.2# ls -lrth
total 8261632
-rw-r-----  1 _mysql  _mysql   2.0G Jun 17 02:43 frag.ibd
-rw-r-----  1 _mysql  _mysql   2.0G Jun 17 03:09 ins_frag.ibd
```

MySQL Table Fragmentation

```
mysql> SELECT
-> table_schema as 'DATABASE',
-> table_name as 'TABLE',
-> CONCAT(ROUND(( data_length + index_length ) / ( 1024 * 1024 * 1024 ), 2), 'G') 'TOTAL',
-> CONCAT(ROUND(data_free / ( 1024 * 1024 * 1024 ), 2), 'G') 'DATAFREE'
-> FROM information_schema.TABLES
-> where table_schema='percona' and table_name='ins_frag';
+----------+----------+-------+----------+
| DATABASE | TABLE.   | TOTAL | DATAFREE |
+----------+----------+-------+----------+
| percona  | ins_frag | 0.00G | 1.96G    |
+----------+----------+-------+----------+
1 row in set (0.01 sec)
```

- Rolling back the INSERT will create the fragmentation.

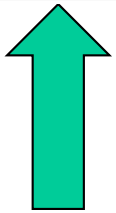
MySQL Table Fragmentation

- Need to rebuild the table to reclaim the disk space.

```
mysql> alter table ins_frag engine=innodb;
Query OK, 0 rows affected (2.63 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
#Linux shell

sakthi-3.2# ls -lrth
total 4131040
-rw-r-----  1 _mysql  _mysql   2.0G Jun 17 02:43 frag.ibd
-rw-r-----  1 _mysql  _mysql   112K Jun 17 03:11 ins_frag.ibd
```



MySQL Table Fragmentation

Case 2: Failed INSERT Statement

Session 1

```
#Linux shell

sakthi-3.2# ls -lrth
total 4131040
-rw-r-----  1 _mysql  _mysql   2.0G Jun 17 02:43 frag.ibd
-rw-r-----  1 _mysql  _mysql   112K Jun 17 04:02 ins_frag.ibd

#MySQL shell

mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into ins_frag select * from frag;   #is running
```

MySQL Table Fragmentation

Session 2

```
mysql> pager grep -i insert ; show processlist;
PAGER set to 'grep -i insert'
| 33 | root            | localhost | percona | Query   |    14 | executing
4 rows in set (0.00 sec)

mysql> kill 33;
Query OK, 0 rows affected (0.00 sec)
```

The INSERT is interrupted and failed.

MySQL Table Fragmentation

Back to Session 1:

```
mysql> insert into ins_frag select * from frag;
ERROR 2013 (HY000): Lost connection to MySQL server during query

#Linux shell

sakthi-3.2# ls -lrth
total 4591616
-rw-r-----  1 _mysql  _mysql   2.0G Jun 17 02:43 frag.ibd
-rw-r-----  1 _mysql  _mysql   212M Jun 17 04:21 ins_frag.ibd

#MySQL shell

mysql> select count(*) from ins_frag;
+----------+
| count(*) |
+----------+
|        0 |
+----------+
1 row in set (0.10 sec)
```

- The INSERT is not completed and there is no data in the table.
- But still, the table .ibd file has grown up to 212M.

MySQL Table Fragmentation

Check the fragmented space through the MySQL client

```
mysql> SELECT
-> table_schema as 'DATABASE',
-> table_name as 'TABLE',
-> CONCAT(ROUND(( data_length + index_length ) / ( 1024 * 1024 ), 2), 'M') 'TOTAL',
-> CONCAT(ROUND(data_free / ( 1024 * 1024 ), 2), 'M') 'DATAFREE'
-> FROM information_schema.TABLES
-> where table_schema='percona' and table_name='ins_frag';
+----------+----------+---------+----------+
| DATABASE | TABLE    | TOTAL   | DATAFREE |
+----------+----------+---------+----------+
| percona  | ins_frag | 0.03M   | 210.56M  |
+----------+----------+---------+----------+
1 row in set (0.01 sec)
```

It shows the table has fragmented space

MySQL Table Fragmentation

Rebuild the table to reclaim the space.

```
mysql> alter table ins_frag engine='innodb';
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

#Linux shell

sakthi-3.2# ls -lrth
total 4131040
-rw-r-----  1 _mysql  _mysql   2.0G Jun 17 02:43 frag.ibd
-rw-r-----  1 _mysql  _mysql   112K Jun 17 04:32 ins_frag.ibd
```

MySQL Table Fragmentation

Case 3: Fragmentation with Page-Splits

created a table with a sorted index (descending)

```
mysql> show create table frag_page_spl\G
*************************** 1. row ***************************
Table: frag_page_spl
Create Table: CREATE TABLE `frag_page_spl` (
`id` int NOT NULL AUTO_INCREMENT,
`name` varchar(16) DEFAULT NULL,
`messages` varchar(600) DEFAULT NULL,
PRIMARY KEY (`id`),
KEY `idx_spl` (`messages` DESC)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
1 row in set (0.07 sec)
```

MySQL Table Fragmentation

- Monitor the page split activity from the table INFORMATION_SCHEMA.INNODB_METRICS
- For this, you need to enable the InnoDB monitor.

```
mysql> SET GLOBAL innodb_monitor_enable=all;
Query OK, 0 rows affected (0.09 sec)
```

- Script created to trigger the INSERTs randomly with 6 parallel threads

MySQL Table Fragmentation

```
mysql> select name,count,type,status,comment from information_schema.innodb_metrics where name like '%index_page_spl%'\G
*************************** 1. row ***************************
name: index_page_splits
count: 52186
type: counter
status: enabled
comment: Number of index page splits
1 row in set (0.05 sec)
```

```
mysql> SELECT
-> table_schema as 'DATABASE',
-> table_name as 'TABLE',
-> CONCAT(ROUND(( data_length + index_length ) / ( 1024 * 1024 ), 2), 'M') 'TOTAL',
-> CONCAT(ROUND(data_free / ( 1024 * 1024 ), 2), 'M') 'DATAFREE'
-> FROM information_schema.TABLES
-> where table_schema='percona' and table_name='frag_page_spl';
+----------+---------------+----------+----------+
| DATABASE | TABLE.        | TOTAL    | DATAFREE |
+----------+---------------+----------+----------+
| percona  | frag_page_spl | 2667.55M | 127.92M  |
+----------+---------------+----------+----------+
1 row in set (0.00 sec)
```

There are **52186 page-splits** operations that occurred, which created **127.92 MB** of fragmentation.