Data aggregation. Real time analytics

Vitalii Melnychuk, Yaware


MySQL®


mongoDB®

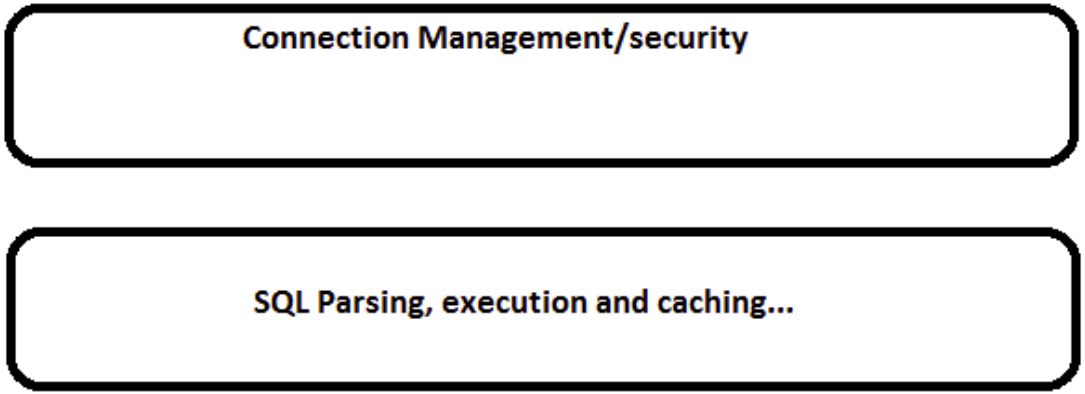

ClickHouse

What we should expecting?

- How each database works? Their benefits.
- How we have to prepare our data?
- Get aggregated data of 4 million rows less than 1 second
- Compare differents type of queries on each database

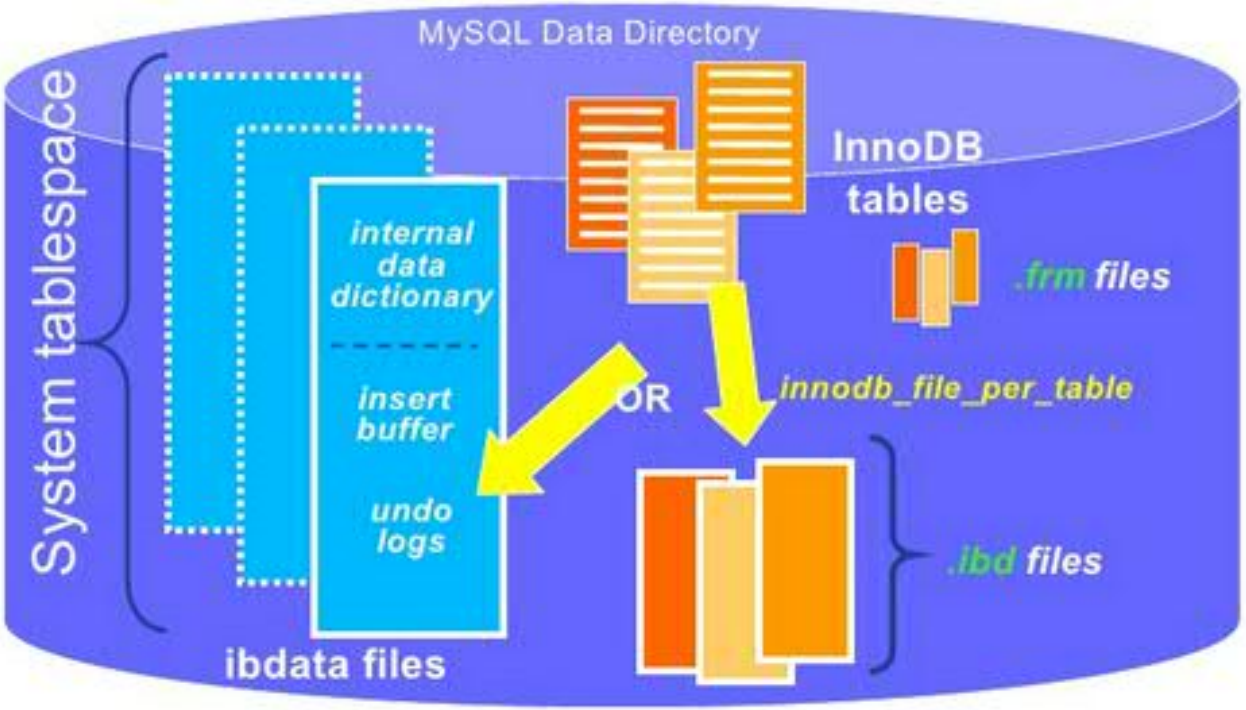
MySQL Architecture

Server's functionality like connection management/authentication... is done in this layer

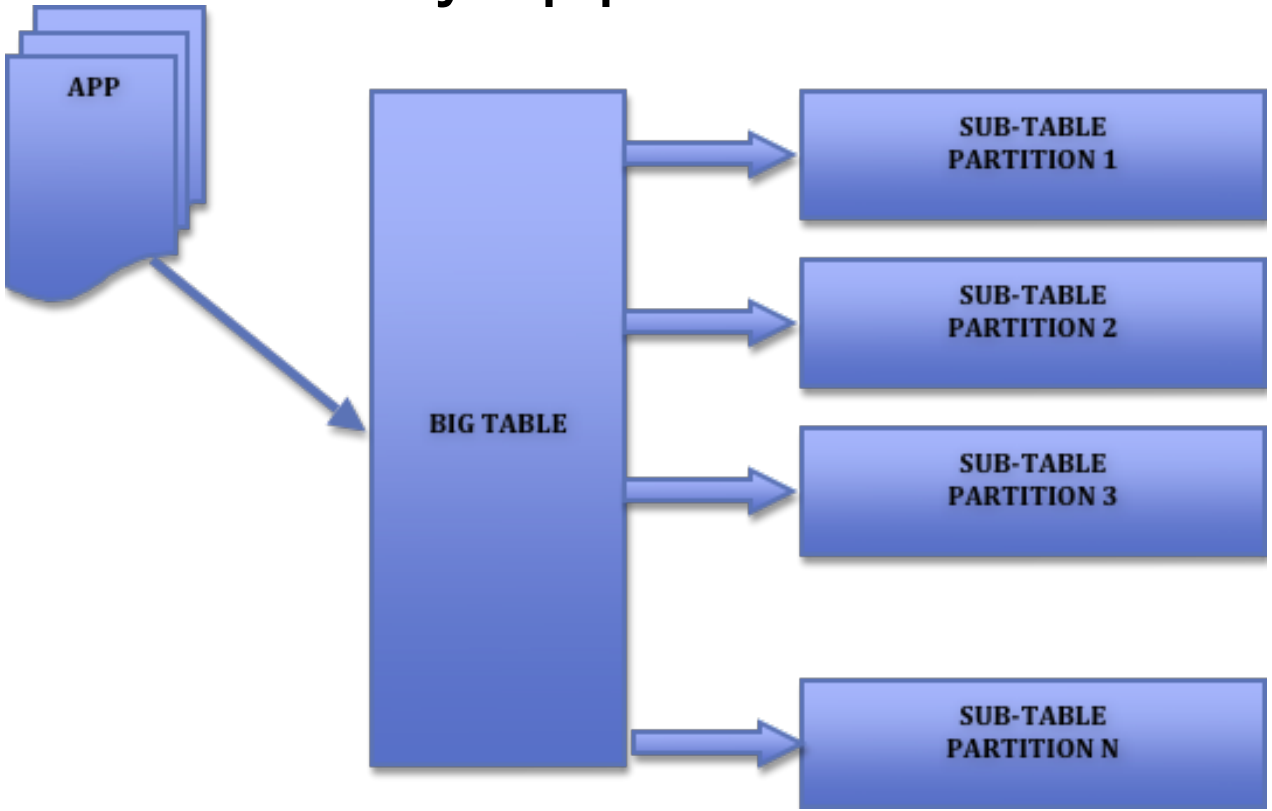


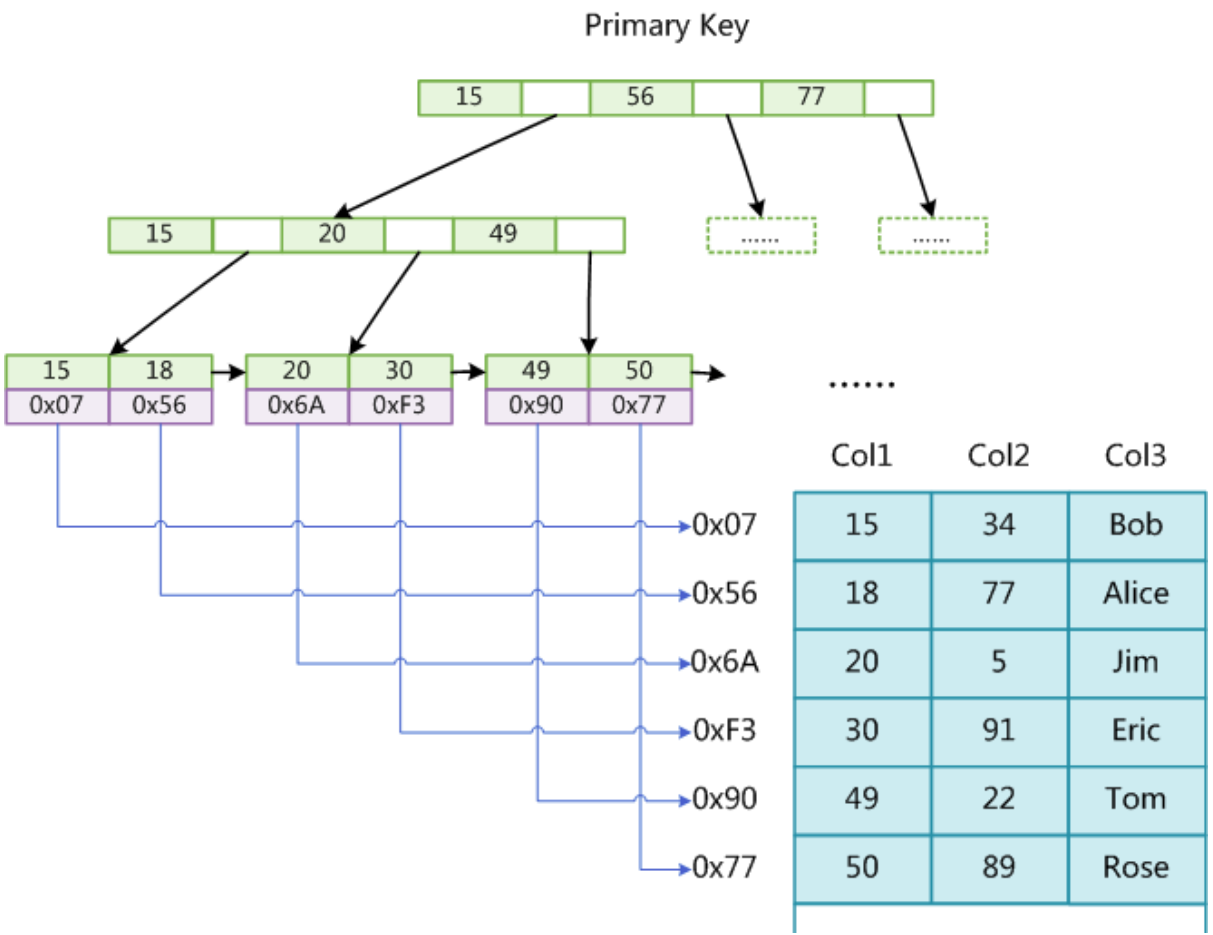
Responsible for storage and retrival of all available information

InnoDB Database Files



Mysql partitions




Primary Key
15
56
77
15
20
49
.....
.....
15
18
20
30
49
50
0x07
0x56
0x6A
0xF3
0x90
0x77
.....
0x07
0x56
0x6A
0xF3
0x90
0x77
Col1
Col2
Col3
15
34
Bob
18
77
Alice
20
5
Jim
30
91
Eric
49
22
Tom
50
89
Rose

Introduction to MySQL triggers

BEFORE INSERT – activated before data is inserted into the table.

AFTER INSERT- activated after data is inserted into the table.

BEFORE UPDATE – activated before data in the table is updated.

AFTER UPDATE - activated after data in the table is updated.

BEFORE DELETE – activated before data is removed from the table.

AFTER DELETE – activated after data is removed from the table.

Approaches of using triggers

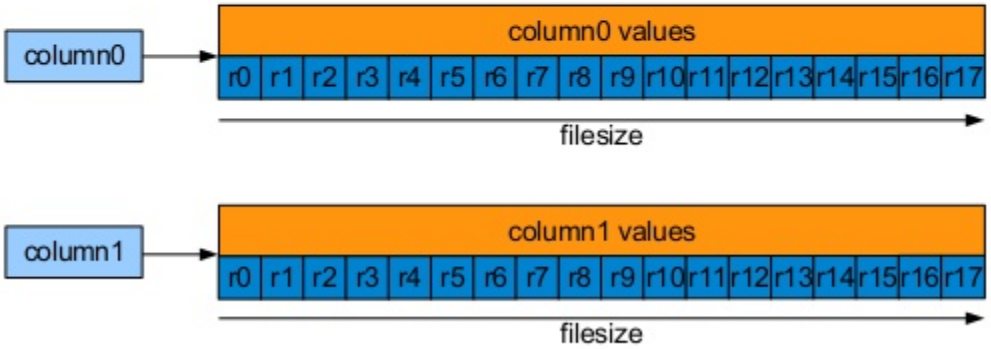
- Log all inserts, updates in your system
- Divide some business functional from server side. Your data will be consistently
- **Aggregate data**

Preparing data

- Crontab
- Triggers
- Add aggregation in server side of project
- Partitions
- Map reduce
- **Query Aggregation**

Column-oriented storage

- Column stores store data in column-specific files
- Simplest case: one datafile per column
- Row values for each column are stored contiguously



ClickHouse

- Parallel processing for single query
- Very fast scans
- SQL support (with limitations)
- Different storage engines (disk storage format)
- Great for structural log/event data as well as time series data (engine MergeTree requires date field)
- Index support (primary key only, not all storage engines)
- Nice command line interface with user-friendly progress bar and formatting

https://habrahabr.ru/company/yandex/blog/303282/

```
SELECT
    count(*),
    toMonth(date) AS mon
FROM wikistat
WHERE (toYear(date) = 2008) AND ((toMonth(da
te) >= 1) AND (toMonth(date) <= 10))
GROUP BY mon
```

count()	mon
2077594099	1
1969757069	2
2081371530	3
2156878512	4
2476890621	5
2526662896	6
2460873213	7
2480356358	8
2522746544	9
2614372352	10

```
10 rows in set. Elapsed: 14.344 sec. Process
ed 23.37 billion rows, 46.74 GB (1.63 billio
n rows/s., 3.26 GB/s.)
```

ClickHouse disadvantages

- No real delete/update support, and no transactions
- No secondary keys
- Own protocol
- Limited SQL support, and the joins implementation is different. If you are migrating from MySQL or Spark, you will probably have to re-write all queries with joins.

sign	applicationId	profileId	productivity	eventDate
1	1	1	10	2017-08-10
-1	1	1	10	2017-08-10
1	1	1	10	2017-08-10
1	2	1	10	2017-08-10
-1	2	1	10	2017-08-10
1	2	1	10	2017-08-10
1	3	1	10	2017-08-10
-1	3	1	10	2017-08-10
1	3	1	10	2017-08-10
1	4	1	-10	2017-08-10
-1	4	1	-10	2017-08-10
1	4	1	-10	2017-08-10
1	5	1	-10	2017-08-10
-1	5	1	-10	2017-08-10
1	5	1	-10	2017-08-10

WHERE profileId = 1

GROUP BY applicationId, profileId
HAVING sum(sign) > 0

How about Clickhouse in production?

- Error handling
- Monitoring
- Migrations
- Performance optimization

Query	ClickHouse	Mysql	Mongo
GROUP BY application_id, EventDate	**0.161 sec**	**1m 24s 540ms**	**1m 5.452s**
application_id, system_user_id, EventDate	**2.000 sec**	**1m 29s 957ms**	**1m 7.003s**
GROUP BY EventDate	**0.128 sec**	**6s 298ms**	**14.084s**
WHERE system_user_id = 3 GROUP BY EventDate	**0.108 sec**	**7s 69ms**	**5.928s**
WHERE toMonth(EventDate) = 4 GROUP BY EventDate	**0.011 sec**	**5s 850ms**	**19.125s**
WHERE toMonth(EventDate) = 4 GROUP BY application_id, system_user_id	**0.067 sec**	**11s 929ms**,	**24.784s**

https://github.com/YawareTeam/clickhouse-php-client

https://habrahabr.ru/company/yandex/blog/303282/

https://github.com/yandex/ClickHouse

https://www.percona.com/blog/2017/02/13/clickhouse-new-opensource-columnar-database/

https://www.percona.com/blog/2017/03/17/column-store-database-benchmarks-mariadb-columnstore-vs-clickhouse-vs-apache-spark/