Mateusz Świerczyński

Piotr Kopka

Group No V

Database performance improvements

1. After creating four different partitions on *AmountPaid* in *Payment* table, two of them have been moved from internal HDD to USB flash drive. In Range Partitioning they were values between 4000 and 6000(one partition) and values above 6000(second partition). In Hash Partitioning they were just names of partitions - Payment_3 and Payment_4.
2. After creating two different partitions on *FlightsNo* in *Passenger* table, one of them has been moved from internal HDD to USB flash drive. In Range Partitioning it was the one containing values below 100. In Hash Partitioning it was just a name - Passenger_1.
3. Created B-Tree index on *departure date* in *Flight* table and after dropping it created Bitmap index
4. Created composite index on *ID, AmountPaid* and *isCardPayment* in *Payment* table.
5. Created Bitmap index on *Brand* in *Airplane* table and after dropping it created B-Tree index.
6. Applied the best solutions from each improvement and measured times of all transactions and the whole database performance.

Improvement / Transaction	Without improvements	Payment (AmountPaid) partitioned by range	Payment (AmountPaid) partitioned by hash	Passenger (FlightsNo) partitioned by range	Passenger (FlightsNo) partitioned by hash	B - Tree index on DepDate in Flight table	Bitmap index on DepDate in Flight table	Composite index on ID, AmountPaid, isCardPayment in Payment table	Bitmap index on Brand in Airplane table	B - Tree index on Brand in Airplane table	With modifications which provided best results
Querying 1	459	389	407	455	397	437	443	391	394	399	411
Querying 2	464	407	416	413	417	441	456	453	449	456	458
Querying 3	414	369	387	414	380	424	391	403	377	382	388
Modyfying 1	536	415	432	425	442	444	450	437	445	450	464
Modyfying 2	478	439	461	455	471	467	479	461	479	473	464
All transactions	2351	2019	2103	2162	2107	2213	2219	2145	2144	2160	2185

Tabel 1. Measured times after applying improvements is database - all times in [ms]

Conclusion

1. Range Partitioning in Payment table significantly improved the time of each transaction. In each case it was better than Hash Partitioning. Conscious division of a table is much better than automatic division done by database management system.
2. In Passenger table, partitioning by Range was as good as automatic division. Hash Partitioning won during data querying and range partitioning gave better results while modifying the data.
3. B-Tree index on Passenger table was better in almost all cases than Bitmap index. B -Tree is useful when the data are highly varied. Departure dates are dates between 2014 and 2017 so they meet this condition.
4. Composite index on Payment table provided better results than the database without improvements but it made only small change of the time of third data querying where, the attributes it contains, were used.
5. Bitmap index on Airplane table was better than B – Tree index. It was expected because there is not so many different airplane brands and in that case Bitmap index is more preferred than B -tree index.
6. Best solutions from each improvement didn't give the best results. The best performance has been achieved after partitioning the whole database on *Payment* table