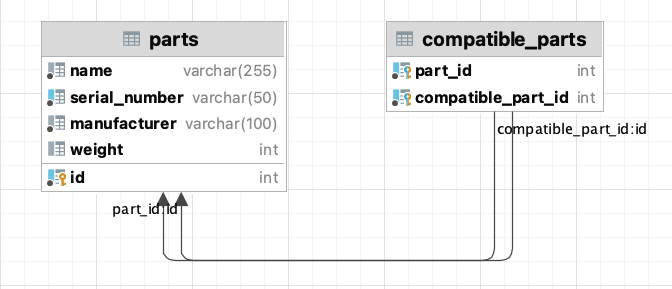
Let’s consider sql and nosql database for storing autoparts.

1. MySQL (relation database)



How does this solution impact if the number of parts in the catalog reaches tens of millions?

* Using Indexing: identify columns used in where clause and create indexes on those columns
* Optimizing query: using EXPLAIN to analyze and optimize the queries.

Partitioning or Sharding will be a challenging task because partitioning/distributing the `compatible\_parts` table is complicated.

1. MongoDB (NoSQL database)

Document looks something like this

{

"\_id": ObjectId("1111..."),

"name": "Auto Part Name",

"serialNumber": "Serial Number",

"manufacturer": "Manufacturer name",

"weight": 100,

"compatibility": [

ObjectId("2222..."),

ObjectId("3333..."),

// ... other ObjectIds

]

}

Having the right indexes, we can have efficient queries. Problem in this design if we want to delete one autopart we need to go through all documents’ compatibly array and update them if they contain the deleted autopart and this operation causes a performance issue in large volume of data like 10 million.

Or if we do not use a reference, but embed compatible parts as object like below

{

"\_id": ObjectId("1111..."),

"name": "Auto Part Name",

"serialNumber": "Serial Number",

"manufacturer": "Manufacturer name",

"weight": 100,

"compatibility": [

{

"name": "Auto Part Name 2",

"serialNumber": "Serial Number 2",

"manufacturer": "Manufacturer name 2",

"weight": 200

},

{

"name": "Auto Part Name 3",

"serialNumber": "Serial Number 3",

"manufacturer": "Manufacturer name 3",

"weight": 300

}

// ... other objects

]

}

In this case we have data redundancy but deleting is not a problem anymore, only update should be solved efficiently.

Horizontally scaling in both MySQL and MongoDB requires work in deciding shard key which making sure data is evenly distributed.

There is also graph based database type, but I do not have experience with it, in the real project I will spend time and do feasibility studies then I will choose the optimal database for this task needs. If partitioning/sharding is solved in the effective way relational database would be good choice.