Indexes:

**Clustered:**

Clustered indexes uses one column as the index which it is sorted by called the key value, it can only be one clustered index per table as you can sort a table by one column only. A table is sorted in an order when it contains a clustered index, if a table do not contain a clustered index, it is called a heap which is an unordered structure. A clustered index is automatically added when a primary key is added, a relational database in the second normal form or higher will contain a clustered index [1].

**Non-clustered:**

A non-clustered index contains the non-clustered index key values and each key value entry has a pointer to the data row that contains the key value. The pointer is called a row locator, there are two types of structures, one for the heap and one for clustered table. For the heap, the row locator points to the row. For a clustered table, it points to the clustered index key [1].

**Expected improvement:**

The estimated execution plan showed that we could create indexes for all out queries except ‘view overdue books’. The impact on these four queries where, view currently on loan (8.67), view not loanable (14.83), look up book by title (99.27) and insert loan (98.74). Based on these estimations, we decided to implement the recommended non-clustered indexes for “view not loanable”, “look up book by title” and “insert loan”. This should improve our system drastically, we decided on not adding a non-clustered index for “view currently on loan” as it only has an impact of 8.67 and is not a query that will be executed a lot, so it is not deemed valuable to use the resources to create a non-clustered index for it.

Based on the estimated execution plan, we expected the “insert loan” to go from 15ms to 2ms, “look up book title” to go from 13ms to 2ms, “view currently on loan” to not change, “view not loanable” to go from 900ms to 755ms and “view overdue books” to not change. As seen in [Figure with execution times] you can see “insert loan” and “look up book by title got reduced by approximately the same as the estimated high 90%. While “view not loanable only got reduced by only 8% when it was estimated to be reduced by 14.83%. Surprisingly, both “view currently on loan” and view overdue books got reduced without adding non-clustered indexes for them, “view currently on loan got reduced 1/5 of the time (20%) and view overdue books got reduced by over 2/5 of the time (43%).

**Difference in execution plans (look up by title):**

In [Figure with look up by title before], it starts with select (Cost: 0%), goes into the nested loops inner join (Cost: 0%), then goes into another nested loops with inner join (Cost: 4%), then does a clustered index scan on Item with primary key (Cost: 96%), goes into another clustered index seek on BookSubject with primary key (Cost: 0%), finally it does a clustered index seek on Book with primary key (Cost: 0%).

In [Figure with look up by title after], it starts with select (Cost: 0%), goes into three nested loops inner join (Cost: 0%), then does a non-cluster index seek on Item wit Title (24%), then uses the key from the non-clustered seek to look up on the table Item (24%), goes into clustered index seek on BookSubject with primary key (Cost: 24%), finally it does a clustered index seek on Book with primary key (Cost: 28%).

Comparing these two tables, it shows adding non-clustered keys save time, before adding the non-clulstered indexes, majority of the resources, 96% was used on the clustered index scan on Item with the primary key that used and the rest was used on the second nested loops before the scan. After adding the indexes, one more loop was added and it found the Item based on the title instead of on the primary key, it now only uses 24% to find the title with the non-clustered index seek and 24% on the clustered key lookup. Adding the indexes also allowed the query to use its resources on the clustered index seek for BookSubject and Book.

1. <https://docs.microsoft.com/en-us/sql/relational-databases/indexes/clustered-and-nonclustered-indexes-described?view=sql-server-2017>