DB 커넥션풀과 HikariCP 란?

문득 SpringBoot로 웹을 개발하면서 궁금증이 생겼습니다.

```
HikariPool-1 - Starting...
HikariPool-1 - Start completed.
```

애는 뭐하는 놈일까?

먼저 HikariPool을 알기 전에 커넥션 비용이라는것이 뭔지 알아야 했습니다.

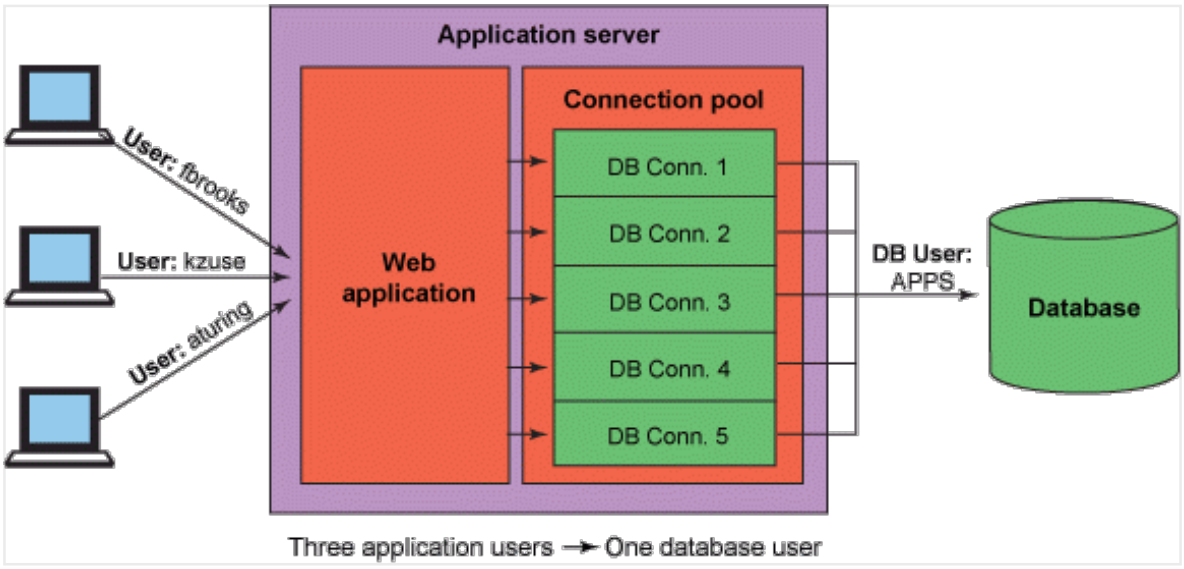
커넥션 비용

WAS(Web Application Server)와 데이터베이스 사이의 연결에는 많은 비용이 듭니다. MySQL 8.0 버전을 기준으로 INSERT 문을 수행할 때 필요한 비용의 비율은 다음과 같습니다.
(괄호의 숫자가 비용의 비율입니다.)

1. Connecting (3)
2. Sending query to server (2)
3. Parsing query (2)
4. Inserting row (1)
5. Inserting index (1)
6. Closing (1)

즉, 서버가 DB를 연결하기 위한 Connecting 비용이 가능 큰 비율을 차지하게 됩니다. 이를 통해 알 수 있는것이 Connection을 생성하는 작업이 가장 큰 비용이 들게 됩니다. 이를 보완하기 위해 만들어진 방법이 커넥션 풀(Connection Pool) 입니다.

커넥션 풀(Connection Pool) 이란?



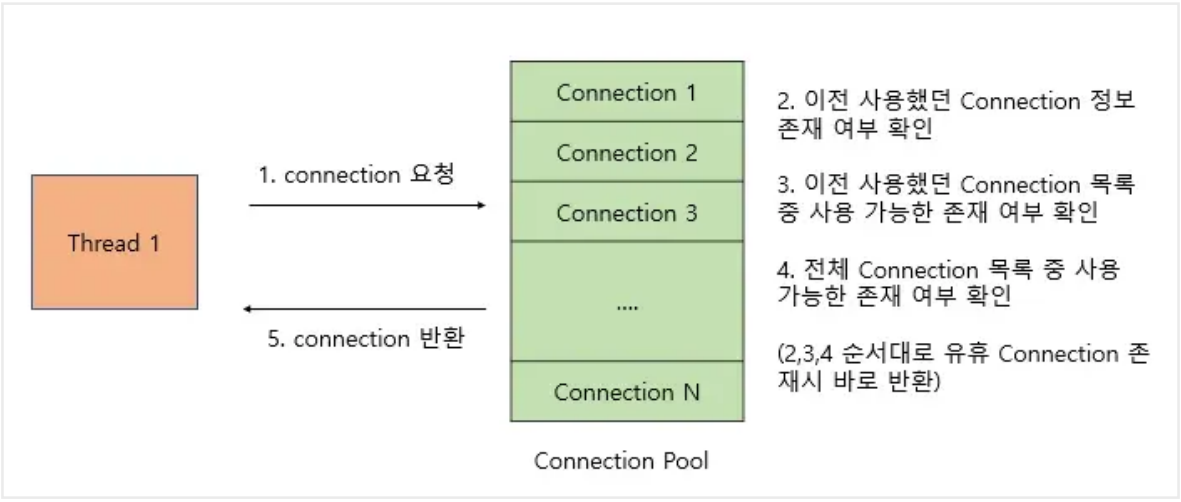
커넥션 풀이란 데이터비스와 연결된 커넥션을 미리 만들어 놓고 이를 Pool로 관리하는 방법입니다. 즉, 필요할 때마다 커넥션 풀의 커넥션을 이용하고 반환하는 기법입니다.
이처럼 만들어 놓은 커넥션을 이용하면 Connection에 필요한 비용이 절감됩니다.

또한 커넥션 풀을 사용하면 커넥션 수를 제한이 가능해서 과도한 접속으로 인한 서버 자원 고갈을 방지 가능합니다. DB접속 모듈을 공통화해 DB 서버의 환경이 바뀔 경우 유지보수를 쉽게 할 수 있습니다.

HikariCP란?

HikariCp는 가벼운 용량과 빠른 속도를 가지는 JDBC의 커넥션 풀 프레임워크입니다. 처음에 봤던 이미지가 바로 HikariCP 입니다. SpringBoot는 커넥션 풀 관리를 위하여 프레임워크인 HikariCP를 사용하는 것입니다.

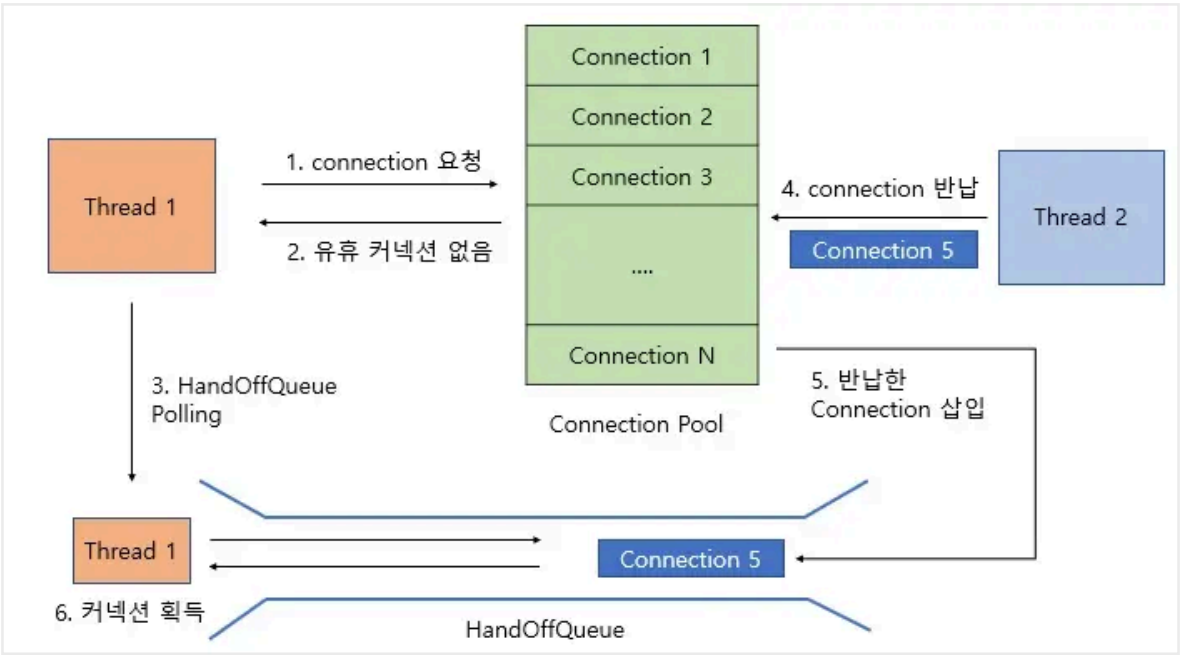
이러한 HikariCP의 동작법을 통해 어떻게 커넥션 풀이 동작하는지 알아보겠습니다.



[그림1] 유휴커넥션이 존재하는 경우

위의 그림과 같이 Thread가 커넥션을 요청하면 유휴 커넥션이 존재한다면 해당 커넥션을 반환해줍니다.

❓ 유휴 : 어떠한 프로그램에 의해서도 사용되지 않는 상태



[그림2] 유휴커넥션이 존재하지 않는 경우

만약 위의 그림처럼 유휴 커넥션이 존재하지 않는다면, HandOffQueue를 Polling하면서 다른 Thread가 커넥션을 반납하기를 기다립니다.
다른 Thread가 커넥션 풀에 커넥션을 반납하면 커넥션 풀은 HandOffQueue에 반납된 커넥션을 삽입하고 HandOffQueue를 Polling 하던 Thread는 커넥션을 획득하게 됩니다.

만약 커넥션 풀의 킥가 너무 작다면, 커넥션을 획드갛기 위해 대기하고 있는

Thread가 많아지게 도리 것입니다. 이러한 문제는 커넥션 풀의 크기를 늘려주면 해결됩니다.

그렇다면 질문이 하나 생기게 됩니다. "커넥션 풀의 크기는 어느정도인게 좋은 것일까?"

커넥션 풀의 크기와 성능

커넥션 풀이 크면 클수록 좋다고 생각하기가 쉽습니다. 생각해보면 커넥션을 많이 가지고 있을 수록 유리 할 것 같기 때문입니다.
하지만 이것은 틀렸습니다. Connection을 사용하는 주체인 Thread의 개수보다 커넥션 풀의 크기가 크다면 사용되지 않고 놀고있는 커넥션이 생기게 되며 메모리 낭비가 됩니다.

MySQL의 공식레페런스를 찾아보면 600여 명의 유저를 대응하는데 15~20개의 커넥션 풀 만으로도 충분하다고 언급합니다. MySQL은 최대 연결 수를 무제한으로 설정한 뒤 부하 테스트를 진행하면서 최적화된 값을 찾는 것을 추천합니다.

우아한 형제들 테크 블로그에서는 이러한 공식을 추천하고 있습니다.

$$pool\ size = Tn \times (Cm - 1) + 1$$

Tn 은 전체 Thread의 개수
Cm 은 하나의 Task에서 동시에 필요한 Connection의 수

결론은 DB와 WAS의 Context Switching 역시 한계가 있기 때문에 Thread Pool의 크기는 Connection Pool의 크기를 결정하는데 매우 중요합니다.

커넥션 풀의 크기와 성능에 대하여 자세히 알고 싶다면 다음 위키를 참고하세요
https://wiki.postgresql.org/wiki/Number_Of_Database_Connections

참고 블로그
https://code-lab1.tistory.com/209#recentEntries
https://dev.mysql.com/doc/refman/8.0/en/insert-optimization.html
https://hyuntaeknote.tistory.com/12
https://code-lab1.tistory.com/209#recentEntries