InnoDB Deadlock Detect Optimization

목적: 이 강의에서는 InnoDB 데드락 감지 기능을 비활성화하여 MySQL 성능을 최적화하는 방법을 배웁니다.

포인트: 데드락 감지 기능이 MySQL 성능에 미치는 영향에 대해 이해합니다.

▼ Understanding Deadlock Detection

데드락 감지 동작: `SELECT ... FOR UPDATE` 와 같은 Lock 을 요구하는 연산은 데드락 감지 스레드를 통해서 데드락을 유발 하는지 검사를 주기적으로 받는다.

데드락 감지가 성능에 미치는 영향: 많은 락을 요구하는 서비스 스레드 처리가 있다면 데드락 감지 스레드 때문에 지연이 발생할 수 있다.

데드락 감지 스레드의 동작 방식: 데드락 감지 스레드가 검사하는 동안에는 잠금을 가진 서비스들은 잠금을 반납하지 못한다.

▼ Optimization Strategies

`innodb_deadlock_detect` **을 비활성화 시키기**: 락을 요구하는 작업이 많다면 이를 비활성화 시키는 것.

- 비활성화하면 데드락이 발생할 수 있으므로 `innodb_lock_wait_timeout` 을 적절한 값으로 설정해야한다. (기본 값: 50)
- 환경에 따라서 Lock 을 기다리는 타임아웃은 조절하는 것이 좋음: (OLTP 환경 VS OLAP 환경)

락을 기다리는 트랜잭션의 유무를 확인하는 법:

- `Performance Schema`
  - `Wait Event` 테이블: 잠금이나 I/O 작업을 통해서 대기하고 있는 스레드에 대한 정보를 확인할 수 있음.
  - `Lock` 테이블: 잠금이 점유되었거나, 잠금을 기다리고 있는 정보를 확인할 수 있음.
- `InnoDB status`
  - InnoDB 내부 상태를 통해서 잠금을 기다리는 트랜잭션을 볼 수 있음.

▼ Practice

Task 1: Try to improve performance by disabling innodb_deadlock_detect

목적: `innodb_deadlock_detect` 설정을 비활성화 시켜서 성능개선을 해보기

시나리오: 잠금을 점유하고 실행하는 쿼리를 동시적으로 실행시켜보고 쿼리 성능을 측정.

```
UPDATE dead_lock_detect_test SET value = 'Updated Value' WHERE id = 375942
```

1. **가상의 테이블 생성:**

```
CREATE TABLE dead_lock_detect_test (
    id INT PRIMARY KEY,
    value VARCHAR(100)
);
```

2. **테스트 데이터 삽입:**

3. **MAX_CONNECTION 증가시키기:**

```
SET GLOBAL MAX_CONNECTIONS = 4000;
```

```
SHOW VARIABLES LIKE 'max_connections';
```

4. `innodb_deadlock_detect` **활성화 시키기:**

```
SET GLOBAL innodb_deadlock_detect = 'ON';
```

```
SHOW VARIABLES LIKE 'innodb_deadlock_detect';
```

5. **FileDescriptor 늘려주기:**

```
ulimit -n 1024
```

6. **mysqlslap 을 통해 락을 요구하는 연산을 동시적으로 보내기:**

```
mysqlslap --no-defaults --host=127.0.0.1 --port=3306 --user=root --password=1234 --query="UPDATE dead_lock_detect_test SET value=no
```

7. `innodb_deadlock_detect` **비활성화 시키기:**

```
SET GLOBAL innodb_deadlock_detect = 'OFF';
```

```
SHOW VARIABLES LIKE 'innodb_deadlock_detect';
```

7. **다시 락을 요구하는 연산을 동시적으로 보내기:**