Auto Increment Lock Mode Optimization

이 강의는 MySQL의 `innodb_autoinc_lock_mode` 설정을 통해 성능을 최적화하는 방법을 다룹니다. 이 설정은 테이블에 `INSERT` 작업을 할 때 동시성 수준을 결정하는 중요한 요소입니다. 이 강의는 이론적 배경과 실습을 통해 학습자가 MySQL 성능 최적화에 필요한 심도 있는 지식을 얻을 수 있도록 구성되었습니다.

▼ Auto Increment Lock Mode Optimization 란?

`Auto Increment` 칼럼을 가진 테이블에 `INSERT` 를 할 때 동시성 수준을 어느정도로 설정할 것인지에 대한 설정이다.

`INSERT` 로 넣을 행의 수가 결정적이냐에 따라서 `INSERT` 의 종류는 크게 두 가지로 나뉜다:

- Simple Inserts:
  - INSERT 로 넣을 행의 수를 미리 알 수 있는 경우.
  - e.g 일반 INSERT 문, Multiple VALUES 를 가진 INSERT 문
- Bulk Inserts:
  - INSERT 로 넣을 행의 수를 미리 알기 어려운 경우.
  - e.g INSERT … SELECT, LOAD DATA, REPLACE … SELECT

그리고 MySQL 은 `innodb_autoinc_lock_mode` 는 총 3가지 종류의 모드가 있다.

각각의 `innodb_autoinc_lock_mode` 는 INSERT 문의 종류에 따라서 동시성 수준이 달라진다.

- `innodb_autoinc_lock_mode = 0` 일 때:
  - 모든 INSERT 문에서 Table Level 의 AUTO-INC Lock 이 사용된다.
- `innodb_autoinc_lock_mode = 1` 일 때:

    - Bulk Insert 에서만 Table Level 의 AUTO-INC Lock 이 사용된다.
- `innodb_autoinc_lock_mode = 2` 일 때:
    - 모든 INSERT 문에서 Table Level 의 AUTO-INC Lock 이 사용되지 않는다. 그보다 가벼운 Mutex 가 사용된다.
    - 이는 Auto Increment 값이 순차적으로 증가함을 보장하지 않는다. Gap 이 발생할 수 있음. 대신에 값이 단조적으로 증가함은 보장할 수 있다.
    - MySQL 8.0 에서의 기본 값이다.

여기서 소개하는 Auto Increment Lock Mode 의 최적화는 `innodb_autoinc_lock_mode = 2` 로 설정하는 것이다.

`innodb_autoinc_lock_mode = 2` 로 설저할 때 주의해야할 것:

- statement 기반의 복제를 사용하고 있다면, Master 와 Slave 의 Auto Increment 칼럼의 값이 다를 수 있다.
- 그래서 Row 기반의 복제를 이용해야한다.

▼ **실습**

목적: 사용자의 장바구니에 있는 항목을 주문으로 전환

출처 테이블: `shopping_cart`

대상 테이블: `orders`

`shopping_cart` 테이블에는 사용자가 선택한 상품의 정보가 담겨 있으며, 이 정보를 `orders` 테이블로 이동시켜 주문을 확정합니다.

장바구니 테이블 설계:

```
CREATE TABLE shopping_cart (
    cart_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    product_id INT NOT NULL,
```

```
    quantity INT NOT NULL,
    cart_added_date DATETIME NOT NULL,
);
```

주문 테이블 설계:

```
CREATE TABLE orders (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    order_date DATETIME NOT NULL,
);
```

장바구니에 들어있는항목을 주문으로 작성하는 Bulk Insert 문 예시:

```
INSERT INTO orders (user_id, product_id, quantity, order_d
SELECT user_id, product_id, quantity, NOW()
FROM shopping_cart
WHERE cart_added_date > '2021-01-01 00:00:00' AND cart_add
```

`shopping_cart_data.csv` **파일을 통해서 장바구니에 테스트 데이터 삽입. OR 다음 Python 코드로 테스트 데이터 생성 후 삽입**

```
import csv
import random
from datetime import datetime, timedelta

# 랜덤 데이터 생성 함수
def generate_random_data(filename, num_records):
    with open(filename, 'w', newline='', encoding='utf-8')
        writer = csv.writer(file)
        writer.writerow(['user_id', 'product_id', 'quantit
```

```
        for _ in range(num_records):
            user_id = random.randint(1, 1000)  # 사용자 ID:
            product_id = random.randint(1, 500)  # 상품 ID:
            quantity = random.randint(1, 10)  # 수량: 1부터
            days_ago = random.randint(1, 60)  # 최대 60일 전
            cart_added_date = datetime.now() - timedelta(d
            cart_added_date_str = cart_added_date.strftime

            writer.writerow([user_id, product_id, quantity

# 데이터 생성 및 CSV 파일 저장
filename = 'shopping_cart_data.csv'
num_records = 100000  # 10000건의 데이터
generate_random_data(filename, num_records)
```

`innodb_autoinc_lock_mode` 값 확인

```
SHOW GLOBAL VARIABLES LIKE 'innodb_autoinc_lock_mode';
```

Python 코드를 통해서 병렬 Bulk Insert 로 주문 데이터 삽입:

```
import pymysql
import time
import threading

# 데이터베이스 설정
host = 'localhost'
port = 3306
user = 'root'
password = '1234'
database = 'test'

# 데이터베이스 연결 설정
def get_db_connection():
    return pymysql.connect(host=host, user=user, password=
```

```
# Bulk Insert 쿼리 실행 함수
def insert_orders():
    conn = get_db_connection()
    try:
        with conn.cursor() as cursor:
            query = """
            INSERT INTO orders (user_id, product_id, quant
            SELECT user_id, product_id, quantity, NOW()
            FROM shopping_cart
            WHERE cart_added_date > '2021-01-01 00:00:00' A
            """
            cursor.execute(query)
        conn.commit()
    finally:
        conn.close()


# 병렬 실행을 위한 스레드 생성 및 실행
def run_parallel_inserts(num_threads):
    threads = []
    for i in range(num_threads):
        thread = threading.Thread(target=insert_orders)
        threads.append(thread)
        thread.start()

    for thread in threads:
        thread.join()

# 스레드 수 설정 및 실행
num_threads = 10  # 동시에 실행할 스레드 수
start_time = time.perf_counter()
run_parallel_inserts(num_threads)
end_time = time.perf_counter()
elapsed_time = end_time - start_time
print(f"Execution time: {elapsed_time} seconds")
```

MySQL 컨테이너 쉘에 접속해서 `innodb_autoinc_lock_mode=1` 로 변경 후 재부팅

1. 컨테이너 쉘에 접속
2. MySQL 서버 설정 파일인 `my.cnf` 파일을 찾는다.
3. my.cnf 파일을 찾고나서 `innodb_autoinc_lock_mode=1` 을 추가한다.
4. `cat` 명령으로 잘 들어갔는지 확인한다.
5. MySQL 컨테이너를 재부팅한다.
6. `innodb_autoinc_lock_mode` 값 확인한다.

```
docker exec -it mysql-container bash
```

```
$ find / -name "my.cnf"
```

```
echo "innodb_autoinc_lock_mode=1" >> my.cnf
```

```
cat my.cnf
```

```
docker restart mysql-container
```

```
SHOW GLOBAL VARIABLES LIKE 'innodb_autoinc_lock_mode';
```

다시 똑같은 Python 코드를 통해서 병렬 Bulk Insert 로 주문 데이터 삽입:

```
import pymysql
import time
import threading

# 데이터베이스 설정
host = 'localhost'
port = 3306
user = 'root'
password = '1234'
database = 'test'
```

```
# 데이터베이스 연결 설정
def get_db_connection():
    return pymysql.connect(host=host, user=user, password=

# Bulk Insert 쿼리 실행 함수
def insert_orders():
    conn = get_db_connection()
    try:
        with conn.cursor() as cursor:
            query = """
            INSERT INTO orders (user_id, product_id, quant
            SELECT user_id, product_id, quantity, NOW()
            FROM shopping_cart
            WHERE cart_added_date > '2021-01-01 00:00:00' A
            """
            cursor.execute(query)
        conn.commit()
    finally:
        conn.close()


# 병렬 실행을 위한 스레드 생성 및 실행
def run_parallel_inserts(num_threads):
    threads = []
    for i in range(num_threads):
        thread = threading.Thread(target=insert_orders)
        threads.append(thread)
        thread.start()

    for thread in threads:
        thread.join()

# 스레드 수 설정 및 실행
num_threads = 10  # 동시에 실행할 스레드 수
start_time = time.perf_counter()
run_parallel_inserts(num_threads)
end_time = time.perf_counter()
```

```
elapsed_time = end_time - start_time
print(f"Execution time: {elapsed_time} seconds")
```

MySQL 서버 설정인 `my.cnf` 파일에서 `innodb_autoinc_lock_mode=1` 을 다시 지우기:

```
sed -i '$ d' my.cnf
```