Multiple-Column Indexes For Optimization

목표: 다중 칼럼 인덱스(multiple-column indexes) 설계 방법을 통한 성능 최적화를 배우게 됩니다.

포인트: 다중 칼럼 인덱스가 무엇이고 설계하는 방법에 대해 이해하는게 중요합니다.

▼ Understanding Multiple-Column Indexes

Multiple-column indexes 는 복합 인덱스 (composite indexes) 라고도 알려져 있으며 이는 테이블의 여러 칼럼을 조합해서 구성되는 인덱스를 말한다.

중요한 건 복합 인덱스는 명시된 칼럼의 순으로 정렬되어 있다는 것이다.

복합 인덱스를 설계할 때 **칼럼 순서에 따라서 쿼리의 성능이 급격하게 차이**가 나기 때문에 올바르게 설계하는 것이 중요하다.

다음과 같이 `(이름, 시력)` 으로 인덱스가 구축된 경우를 보자:

- 먼저 이름 순대로 정렬되어 있고 이후에 시력 순대로 정렬되어 있다.
- 그러므로 이름이 먼저 정렬되어 있기 때문에 `시력` 만으로 쿼리를 날릴 경우에 인덱스 풀스캔이 발생할 것이다.

이름	시력
Alice	20/20
Alice	20/30
Alice	20/40
Bob	20/20
Bob	20/30
Bob	20/40
Charlie	20/20
Charlie	20/30
Charlie	20/40

▼ Effective Design Strategies

일반적인 복합 인덱스를 설계하는 기준은 카디널리티 (Cardinality) 이다.

카디널리티가 높은 칼럼을 복합 인덱스의 선행 칼럼으로 두는 것이다.

중요한 건 카디널리티만 고려해서 인덱스를 설계하면 안된다는 것이다. 이것 말고도 고려할 사항은 많다:

- 자주 사용하는 쿼리가 무엇인가?
- 조인에도 인덱스가 사용하는가?:
- 인덱스의 선행 칼럼이 범위 기반의 쿼리로 많이 이용되는가?

- 이렇게 인덱스를 설계하면 슬로우 쿼리가 발생하지 않을까?

잠깐! 칼럼의 카디널리티는 어떻게 확인할 수 있을까?

- 이미 생성된 인덱스라면 인덱스 통계 테이블 (= `innodb_index_stats`)을 통해서 볼 수 있다.
- 아직 인덱스를 생성하지 않았더라면 MySQL 8.0 에서는 히스토그램을 통해서 확인할 수 있다.

```
# 히스토그램을 이용해서 조회하는 방법
ANALYZE TABLE your_table UPDATE HISTOGRAM ON your_column WITH 100 BUCKETS;
```

▼ Too many columns in a composite index hinder performance

복합 인덱스가 너무 많은 칼럼으로 구성되어 있어서 성능이 잘 나오지 않으면 어떻게 해야할까?

여러 칼럼을 조합해서 만든 `hashed` 칼럼을 인덱스로 만들어서 이를 극복할 수 있다.

```
SELECT * FROM tbl_name
WHERE hash_col=MD5(CONCAT(val1,val2))
AND col1=val1 AND col2=val2;
```

▼ Practice

Task 1: Impact of Well-Designed Indexes

목적: 다중 칼럼 인덱스 설계를 해보고, 잘못 설계된 경우와 성능 차이를 비교

시나리오: 온라인 쇼핑몰의 주문 관리 시스템

테이블 구조: `orders`

- `order_id` (주문 ID, 기본 키)
- `customer_id` (고객 ID)
- `order_date` (주문 날짜)
- `product_id` (제품 ID)
- `quantity` (수량)
- `status` (주문 상태)

어플리케이션에서 발생하는 쿼리 유형:

1. **특정 고객의 특정 날 부터 최근 주문 조회:**
   - `SELECT * FROM orders WHERE customer_id = X AND order_date > Y ORDER BY order_date DESC LIMIT 10;`
2. **특정 고객의 특정 상태 주문 조회**:
   - `SELECT * FROM orders WHERE customer_id = X AND status = '배송 완료';`
3. **장기간 구매하지 않은 고객 목록**:
   - `SELECT customer_id FROM orders WHERE order_date < '2023-01-01' customer_id`

복합 인덱스

- `customer_id` 와 `order_date` : 고객별 최근 주문 조회에 최적화
- `customer_id` 와 `status` : 특정 고객의 특정 상태 주문 조회에 최적화
- `order_date` **:** 장기간 구매하지 않은 고객 목록에 적합

Task 2: Hashed Column Index

목적: 다중 칼럼 인덱스를 `Hashed` 칼럼을 가진 인덱스로 변경해보고 두 쿼리의 성능을 비교해보기

시나리오: 온라인 쇼핑몰의 주문 관리 시스템

1. **주문 테이블 생성:**

```
CREATE TABLE orders (
    order_id INT AUTO_INCREMENT,
    customer_id INT,
    order_date DATE,
    product_id INT,
    quantity INT,
    status VARCHAR(50),
    PRIMARY KEY (order_id)
);
```

2. **테스트 데이터 삽입:**

3. **다중 칼럼 인덱스 생성:**

```
CREATE INDEX idx_customer_product_status ON orders(customer_id, product_id, status);
```

4. **다중 칼럼 인덱스를 사용하는 SELECT 문과 실행 계획:**

```
SELECT * FROM orders
WHERE customer_id = 3905
  AND product_id = 968
  AND status = '배송 완료'
  AND order_date > '2022-01-01';
```

```
EXPLAIN SELECT * FROM orders
WHERE customer_id = 3905
  AND product_id = 968
  AND status = '배송 완료'
  AND order_date > '2022-01-01';
```

```
EXPLAIN ANALYZE SELECT * FROM orders
WHERE customer_id = 3597
  AND product_id = 233
  AND status = '배송 완료'
  AND order_date > '2022-01-01';
```

5. `hashed` **칼럼을 생성하고 인덱스로 추가:**

```
ALTER TABLE orders ADD COLUMN order_product_status_hash CHAR(32) AS (MD5(CONCAT(customer_id, '-', product_id, '-', status))) STORED
```

```
CREATE INDEX idx_order_product_status_hash ON orders(order_product_status_hash);
```

6. `hashed` 칼럼을 가진 인덱스를 사용하는 SELECT 문과 실행 계획:

```
SELECT * FROM orders
WHERE order_product_status_hash = MD5(CONCAT('3905', '-', '968', '-', '배송 완료'))
  AND customer_id = 3905
  AND product_id = 968
  AND status = '배송 완료'
  AND order_date > '2022-01-01';
```

```
EXPLAIN SELECT * FROM orders
WHERE order_product_status_hash = MD5(CONCAT('3905', '-', '968', '-', '배송 완료'))
  AND customer_id = 3905
  AND product_id = 968
  AND status = '배송 완료'
  AND order_date > '2022-01-01';
```

```
EXPLAIN ANALYZE SELECT * FROM orders
WHERE order_product_status_hash = MD5(CONCAT('3905', '-', '968', '-', '배송 완료'))
  AND customer_id = 3597
  AND product_id = 233
  AND status = '배송 완료'
  AND order_date > '2022-01-01';
```