Index Dive Optimizaiton

목표: 이 강의에서는 MySQL의 Index Dive 최적화 방법을 배우게 됩니다.

포인트: Index Dive의 개념과 실행 계획 수립에서의 역할을 이해하고, 이를 최적화하는 방법을 학습합니다.

▼ Understanding Index Dive

Index Dive란: MySQL이 실행 계획을 수립할 때 사용 가능한 인덱스를 평가하는 과정.

- 실제 인덱스 검색에 사용할 데이터를 가지고 스캔을 조금 해본다.
- 이 과정을 통해서 수집한 정보를 바탕으로 사용할 인덱스를 결정한다.

Index Dive 예시:

- `col_name IN (10, 20, 30)` 에서는 10, 20, 30 총 3가지 범위가 있음
- `col_name = 10` 을 Index Dive 를 보자.
  - `col_name = 10` 에서 사용할 수 있는 인덱스를 결정한다.
  - `col_name = 10`이라는 조건에 해당하는 범위에서 첫 번째 다이빙이 이루어진다.
  - 이 다이빙은 해당 값에 대한 인덱스의 시작 부분을 탐색한다.
  - 두 번째 다이빙은 같은 값(10)에 대한 인덱스의 끝 부분을 탐색한다.
  - 이 두 다이빙을 통해 `col_name = 10` 조건을 만족하는 행의 수가 추정된다.
- 정리하면 각 범위당 두 번의 Index Dive 를 함: (인덱스의 시작과 끝)

Index Dive의 역할: 효율적인 쿼리 실행을 위한 인덱스 선택에 중요한 역할을 함.

MySQL 옵티마이저의 실행 계획 우선순위:

- 1. Index Dive 를 이용한 예측:
- 2. 히스토그램을 이용한 에측
- 3. 인덱스 통계를 이용한 에측

Index Dive 의 비용이 커지는 경우:

- `IN(List)` 절의 `List` 에 많은 요소가 있는 경우
- `IN(List)` 절과 `OR` 절을 같이 여러개 쓰는 경우
- 많은 범위를 다루는 경우: (e.g `SELECT c1 FROM t1  WHERE ( c1 > 10 AND  c1 < 11) OR ... <200 clauses>;`)

Index Dive 과정을 Skip 할 수 있는 경우 (아래의 조건이 일부 만족하는 경우):

- 조인이 없는 단일 테이블에 대한 쿼리
- 단일 인덱스에 대한 FORCE INDEX 힌트를 준 경우
- FULLTEXT 인덱스가 아니면서 인덱스가 Unique Constarint 가 걸려있지 않은 경우
- 서브쿼리가 없을 것:
- DISTINCT, GROUP BY, ORDER BY 절이 없을 것
- (MySQL 8.0 이전에는 `eq_range_index_dive_limit` 변수를 통해서만 Index DIve 를 Skip 할 수 있었음.)

Index Dive 를 못하게 만드는 시스템 변수: `range_optimizer_max_mem_size`

- `range_optimizer_max_mem_size` 로 지정한 메모리 값보다 더 많은 `IN (List)` 의 요소들이 온다면 해당 실행 계획은 포기되며, 인덱스는 사용되지 않는다.

▼ Optimizing Index Dive

`FORCE INDEX` **사용하기:**

- 지정한 Index 로 스캔을 하도록 만들어서 Index Dive 를 피한다.
- 쿼리에 적합한 인덱스인지 검토가 필요하다.

`eq_range_index_dive_limit` **변수 조정 하기:**

- `eq_range_index_dive_limit` 변수로 지정한 값보다 많은 비교를 해야한다면 Index Dive 는 하지 않고 인덱스 통계 정보만을 가지고 실행 계획을 수립
- '범위 비교' 가 아닌 '동등 비교' 에 해당함.

▼ Practice

Task 1: Optimize Index Dive using FORCE INDEX

목적: Index DIve 비용이 높은 쿼리를 작성해보고, `FORCE INDEX` 를 이용해서 비용 최적화하기

시나리오: 다수의 고객이 구매한 주문을 동시에 확인

테이블: `orders`

- `id` (기본 키)
- `customer_id` (고객 식별자)
- `order_date` (주문 날짜)
- `total_amount` (주문 총 금액)

1. **Index Dive 를 보기 위해** `performance_schema` **가 활성화 되었는지 확인:**

```
SHOW VARIABLES LIKE 'performance_schema';
```

```
UPDATE performance_schema.setup_instruments SET ENABLED = 'YES', TIMED = 'YES' WHERE NAME LIKE 'statement/%';
UPDATE performance_schema.setup_instruments SET ENABLED = 'YES', TIMED = 'YES' WHERE NAME LIKE 'stage/%';
UPDATE performance_schema.setup_consumers SET ENABLED = 'YES' WHERE NAME LIKE 'events_stages%';
UPDATE performance_schema.setup_consumers SET ENABLED = 'YES' WHERE NAME LIKE 'events_statements%%';
```

2. **테이블 설계**:

```
CREATE TABLE orders (
  id INT AUTO_INCREMENT PRIMARY KEY,
  customer_id INT NOT NULL,
  order_date DATE NOT NULL,
  total_amount DECIMAL(10, 2) NOT NULL
);
```

3. **Index Dive 를 수행하는 인덱스 설계:**

```
CREATE INDEX idx_customer_order ON orders(customer_id, order_date);
```

4. **테스트 데이터 삽입:**

5. **현재 세션에서 `eq_range_index_dive_limit` 값 확인:**

```
SHOW SESSION VARIABLES LIKE 'eq_range_index_dive_limit';
```

6. **Index Dive 비용이 있는 쿼리 실행:**

```
SELECT *
FROM orders
WHERE customer_id IN (101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122,
```

7. **`FORCE INDEX` 를 통해 Index Dive 비용을 무시하도록 하고 실행:**

```
SELECT *
FROM orders FORCE INDEX (idx_customer_order)
WHERE customer_id IN (101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122,
```

8. **`performance_schema` 를 통해서 성능 비교:**

```
SELECT stages.EVENT_ID, statements.EVENT_ID, statements.END_EVENT_ID, statements.SQL_TEXT, stages.EVENT_NAME, stages.TIMER_WAIT/100
FROM performance_schema.events_stages_history_long AS stages
JOIN performance_schema.events_statements_history_long AS statements
ON (stages.EVENT_ID >= statements.EVENT_ID AND stages.EVENT_ID <= statements.END_EVENT_ID)
WHERE stages.EVENT_NAME LIKE '%statistics%'
  AND statements.SQL_TEXT LIKE '%FROM orders%'
  AND statements.SQL_TEXT NOT LIKE '%SELECT stages.EVENT_ID,%'
ORDER BY statements.EVENT_ID DESC;
```

9. **동등 비교를 더 추가해서 Index Dive 비용 보기:**

```
SELECT *
FROM orders
WHERE customer_id IN (101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122,
```

```
SELECT *
FROM orders FORCE INDEX (idx_customer_order)
WHERE customer_id IN (101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122,
```

10. **`eq_range_index_dive_limit` 값을 초과한 동등 비교를 통해서 Index DIve 활용되지 않는 것 확인:**

```
SELECT *
FROM orders
WHERE customer_id IN (101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122,
```

```
SELECT *
FROM orders FORCE INDEX (idx_customer_order)
```

```
WHERE customer_id IN (101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122,
```