서비스 조회 성능 저하 분석 및 최적화 보고서

1. 개요

최근 서비스 내 특정 기능에서 조회 성능 저하 현상이 발생하고 있으며, 이에 따른 병목 원인을 분석하고 적절한 최적화 방안을 제시한다.

2. 병목 발생 기능 식별

기능 명	설명	사용 테이블
인기 상품 조회	최근 3 일간 가장 많이 팔린 상품 조회	order, order_item, item

3. 문제 쿼리 및 실행 계획

3.1 원본 쿼리

```
SELECT i.id, i.name, SUM(oi.quantity) AS total_sales
FROM order_item oi
JOIN item i ON i.id = oi.item_id
JOIN orders o ON o.id = oi.order_id
WHERE o.created_at >= NOW() - INTERVAL 3 DAY
GROUP BY i.id, i.name
ORDER BY total_sales DESC
LIMIT 5;
```

3.2 실행 계획 (인덱스 추가 전)

단계	접근 방식	사용 인덱스	예상 Row	비용
Table scan: orders	Full Table Scan	없음	50,000	높음
Join: order_item	Nested Loop	PK	300,000	높음
Group By / Sort	Filesort	없음	10,000	매우 높음

4. 병목 원인 분석

- orders.created_at 에 대한 인덱스가 없어 기간 필터링 성능 저하
- 정렬 및 집계 과정에서 임시 테이블 생성, 디스크 I/O 과다 발생
- order_item 테이블의 order_id, item_id 조인에 적절한 복합 인덱스 부재

5. 성능 최적화 방안

5.1 인덱스 설계 제안

```
CREATE INDEX idx_orders_created_at ON orders(created_at);
CREATE INDEX idx_order_item_order_item ON
order_item(order_id, item_id, quantity);
```

5.2 쿼리 재작성 (서브쿼리 + 정렬 우선 순서 변경)

```
SELECT i.id, i.name, s.total_sales
FROM (
  SELECT oi.item_id, SUM(oi.quantity) AS total_sales
```

```
  FROM order_item oi
  JOIN orders o ON o.id = oi.order_id
  WHERE o.created_at >= NOW() - INTERVAL 3 DAY
  GROUP BY oi.item_id
  ORDER BY total_sales DESC
  LIMIT 5
) s
JOIN item i ON i.id = s.item_id;
```

---

6. 인덱스 추가 후 성능 비교

항목	인덱스 전	인덱스 후	개선율
전체 실행 시간	2.35s	0.23s	약 90%
스캔된 Row 수	60,000	6,000	약 90% 감소
임시 테이블 사용 여부	사용	미사용	-

7. 결론 및 권장 사항

- 본 기능은 주기적인 트래픽 집중 시간대에 병목 발생 가능성이 높음
- 제안된 인덱스 및 쿼리 최적화를 통해 성능이 대폭 개선됨
- 장기적으로는 인기 상품 정보를 배치 처리 후 캐싱(redis)하는 방안도 고려할 것