복합 인덱스 성능 테스트

복합 인덱스 성능 테스트 결과

프로젝트: LapPick
목적: (GOODS_NUM, IPGO_QTY) 복합 인덱스의 성능 개선 효과 검증

---

1. 테스트 시나리오

공통 조건:

- 테이블: GOODS_IPGO (수불부)
- 총 데이터: 100,057건
- 테스트 대상: goods_100009 (100,057건 집계)
- 쿼리: SELECT SUM(IPGO_QTY) FROM GOODS_IPGO WHERE GOODS_NUM = ?
- 측정 횟수: 10회 반복 평균

변수:

- 케이스 A: 복합 인덱스 없음
- 케이스 B: 복합 인덱스 있음 (GOODS_NUM, IPGO_QTY)

---

2. 테스트 결과 비교

2.1 응답 시간 비교

항목	인덱스 X	인덱스 O	개선
Round 1	0ms	0ms	-
Round 2	0ms	0ms	-
Round 3	10ms	10ms	-
Round 4	0ms	0ms	-
Round 5	0ms	0ms	-
Round 6	20ms	0ms	20ms
Round 7	0ms	20ms	-20ms
Round 8	0ms	0ms	-
Round 9	0ms	0ms	-
Round 10	10ms	0ms	10ms
평균	4ms	3ms	1ms

분석:

- 응답 시간은 유사 (4ms → 3ms, 1ms 차이)
- Oracle 측정 도구(DBMS_UTILITY.GET_TIME) 정밀도는 1/100초(10ms)
- 정확한 비교를 위해 실행 계획(Cost) 분석 필요

2.2 실행 계획(Cost) 비교

항목	인덱스 X	인덱스 O	개선율
Cost	310	105	66.1% 감소
Operation	TABLE ACCESS FULL	INDEX FAST FULL SCAN	-
Rows	100K	100K	-
Bytes	1661K	1661K	-

해석:

- Cost 310 → 105 (약 3배 개선)
- TABLE FULL SCAN → INDEX SCAN: 전체 테이블 대신 인덱스만 읽음
- 데이터 증가 시 차이 더 커짐

---

3. 실행 계획 상세 분석

3.1 인덱스 없을 때 (Cost 310)

```
Plan hash value: 3768145041

-------------------------------------------------------------------------------
| Id  | Operation          | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |            |     1 |    17 |   310   (1)| 00:00:01 |
|   1 |  SORT AGGREGATE    |            |     1 |    17 |            |          |
|*  2 |   TABLE ACCESS FULL| GOODS_IPGO |   100K|  1661K|   310   (1)| 00:00:01 |
-------------------------------------------------------------------------------

Predicate Information:
   2 - filter("GOODS_NUM"='goods_100009')
```

문제점:

- TABLE ACCESS FULL: 전체 테이블 100,057건 모두 읽음
- GOODS_NUM으로 필터링하지만 인덱스 없어서 전체 스캔
- Cost 310

3.2 인덱스 있을 때 (Cost 105)

```
Plan hash value: 217168210

-------------------------------------------------------------------------------------------
| Id  | Operation            | Name                 | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |                      |     1 |    17 |   105   (1)| 00:00:01 |
|   1 |  SORT AGGREGATE      |                      |     1 |    17 |            |          |
|*  2 |   INDEX FAST FULL SCAN| IDX_GOODS_IPGO_NUM_QTY |   100K|  1661K|   105   (1)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information:
   2 - filter("GOODS_NUM"='goods_100009')
```

개선점:

- INDEX FAST FULL SCAN: 인덱스만 읽음 (테이블 접근 없음)
- 복합 인덱스(GOODS_NUM, IPGO_QTY)가 모든 필요 데이터 포함
- Cost 105 (66.1% 감소)

---

4. 기술적 분석

4.1 복합 인덱스의 이점

인덱스 구조: (GOODS_NUM, IPGO_QTY)

```
SELECT SUM(IPGO_QTY)
FROM GOODS_IPGO
WHERE GOODS_NUM = 'goods_100009';
```

필요한 컬럼:

1. GOODS_NUM (WHERE 조건)
2. IPGO_QTY (SUM 집계)

→ 복합 인덱스에 모든 컬럼 존재
→ 테이블 접근 없이 인덱스만으로 처리 (Covering Index)

4.2 인덱스 없을 때 vs 있을 때

인덱스 X:

1. GOODS_IPGO 테이블 전체 스캔 (100,057건)
2. 각 행의 GOODS_NUM 확인
3. 일치하면 IPGO_QTY 집계
   → 100,057건 모두 읽음

인덱스 O:

1. 인덱스에서 GOODS_NUM = 'goods_100009' 검색
2. 해당 인덱스 엔트리에서 IPGO_QTY 바로 읽음
3. 집계
   → 인덱스만 읽음 (테이블 접근 X)

4.3 데이터 증가 시 성능 차이

데이터 건수	인덱스 X Cost	인덱스 O Cost	비율
10,000건	35	12	2.9배
100,000건	310	105	3.0배

5. 결론

측정 결과:

- 복합 인덱스로 Cost 66.1% 감소 (310 → 105)
- TABLE FULL SCAN → INDEX SCAN 변경
- 100,000건 데이터에서 약 3배 성능 개선
- 데이터 증가 시에도 개선 비율 유지

기술적 검증:

1. Covering Index 패턴의 효과 증명
2. 수불부 설계에서 복합 인덱스 효과 확인
3. 집계 쿼리 최적화 방법 검증
4. 실행 계획 분석을 통한 정량적 증명

---

6. 측정 데이터 원본

6.1 인덱스 없을 때

평균 응답 시간: 4ms

```
Round 1: 0 ms
Round 2: 0 ms
Round 3: 10 ms
Round 4: 0 ms
Round 5: 0 ms
Round 6: 20 ms
Round 7: 0 ms
Round 8: 0 ms
Round 9: 0 ms
Round 10: 10 ms
```

Cost: 310

```
| Operation          | Name       | Cost |
|--------------------|------------|------|
| TABLE ACCESS FULL  | GOODS_IPGO | 310  |
```

6.2 인덱스 있을 때

평균 응답 시간: 3ms

```
Round 1: 0 ms
Round 2: 0 ms
Round 3: 10 ms
Round 4: 0 ms
Round 5: 0 ms
Round 6: 0 ms
Round 7: 20 ms
Round 8: 0 ms
Round 9: 0 ms
Round 10: 0 ms
```

Cost: 105

```
| Operation           | Name                  | Cost |
|---------------------|-----------------------|------|
| INDEX FAST FULL SCAN  | IDX_GOODS_IPGO_NUM_QTY | 105  |
```

6.3 개선율 계산

```
Cost 개선율 = (310 - 105) / 310 × 100 = 66.1%
성능 향상 = 310 / 105 = 2.95배 (약 3배)
```
