Top N 쿼리 개념서

이 문서는 SQLD 시험을 위한 Top N 쿼리의 개념과 구현 방식을 다룹니다. Top N 쿼리의 기본 개념, DBMS별 구현 방식, 고급 활용, 성능 최적화, 주의사항 및 실전 활용 예제를 포함하고 있습니다. 또한 SQLD 시험 대비를 위한 팁도 제공합니다.

1. Top N 쿼리의 기본 개념

1.1 정의

- 데이터를 정렬한 후 상위 N개의 행을 추출하는 쿼리
- 특정 조건에 맞는 학습/워크숍의 데이터를 조회할 때 사용
- DBMS별로 구현 방식이 다름

1.2 주요 용도

1. **순위 기반 조회**
   - 매출액 상위 N개 지점
   - 성적 상위 N명의 학생
2. **최신/최근 데이터 조회**
   - 최근 N일간의 주문 내역
   - 최신 게시글 N개

2. DBMS별 구현 방식

2.1 Oracle

- **ROWNUM 사용**

```
-- 기본 형식
SELECT *
FROM (
  SELECT *
  FROM 테이블
  ORDER BY 컬럼
) WHERE ROWNUM <= N;

-- 실제 예제
SELECT *
FROM (
  SELECT *
  FROM 사원
  ORDER BY 급여 DESC
) WHERE ROWNUM <= 5;
```

- **ROW_NUMBER() 사용**

```
SELECT *
FROM (
  SELECT 사원.*,
      ROW_NUMBER() OVER (ORDER BY 급여 DESC) as rn
  FROM 사원
) WHERE rn <= 5;
```

2.2 SQL Server

- **TOP 절 사용**

```
-- 기본 형식
SELECT TOP N *
FROM 테이블
ORDER BY 컬럼;

-- 실제 예제
SELECT TOP 5 *
FROM 사원
ORDER BY 급여 DESC;

-- 동률 처리 (WITH TIES)
SELECT TOP 5 WITH TIES *
FROM 사원
ORDER BY 급여 DESC;
```

- **ROW_NUMBER() 사용**

```
SELECT *
FROM (
  SELECT *
      ROW_NUMBER() OVER (ORDER BY 급여 DESC) as rn
  FROM 사원
) t
WHERE rn <= 5;
```

2.3 MySQL/MariaDB

- **LIMIT 절 사용**

```
-- 기본 형식
SELECT *
FROM 테이블
ORDER BY 컬럼
LIMIT N;

-- 실제 예제
SELECT *
FROM 사원
ORDER BY 급여 DESC
LIMIT 5;

-- OFFSET 사용
SELECT *
FROM 사원
ORDER BY 급여 DESC
LIMIT 5 OFFSET 2;  -- 3번째부터 5개
```

3. Top N 쿼리의 고급 활용

3.1 그룹별 Top N

```
-- 부서별 급여 상위 3명
SELECT *
FROM (
  SELECT 사원.*,
      ROW_NUMBER() OVER (PARTITION BY 부서번호
                         ORDER BY 급여 DESC) as rn
  FROM 사원
) WHERE rn <= 3;
```

3.2 동률 처리

```
-- RANK 사용
SELECT *
FROM (
  SELECT 사원.*,
      RANK() OVER (ORDER BY 급여 DESC) as rnk
  FROM 사원
) WHERE rnk <= 5;

-- DENSE_RANK 사용
SELECT *
FROM (
  SELECT 사원.*,
      DENSE_RANK() OVER (ORDER BY 급여 DESC) as drnk
  FROM 사원
) WHERE drnk <= 5;
```

3.3 조건부 Top N

```
-- 2023년 입사자 중 급여 상위 5명
SELECT *
FROM (
  SELECT *
  FROM 사원
  WHERE EXTRACT(YEAR FROM 입사일) = 2023
  ORDER BY 급여 DESC
) WHERE ROWNUM <= 5;
```

4. 성능 최적화

4.1 인덱스 활용

```
-- 인덱스 생성
CREATE INDEX idx_emp_sal ON 사원(급여 DESC);

-- 인덱스를 활용한 쿼리
SELECT *
FROM (
  SELECT /*+ INDEX(사원 idx_emp_sal) */ *
  FROM 사원
) WHERE ROWNUM <= 5;
```

4.2 실행 계획 고려사항

1. **정렬 작업 최소화**
   - 인덱스를 통한 정렬 활용
   - 필요한 컬럼만 선택
2. **중간 결과 집합 크기**
   - 필터링을 먼저 수행
   - 불필요한 정렬 작업 제거

5. 주의사항 및 제약사항

5.1 일반적인 주의사항

1. **ORDER BY 필수**
   - 정렬 없는 Top N은 의미 없음
   - 명시적 정렬 조건 지정
2. **인덱스 활용**
   - 정렬 후 필터 제한
   - 잘못된 결과 방지

5.2 DBMS별 제약사항

1. **Oracle**
   - ROWNUM은 WHERE절에서 사용 시 주의
   - 1부터 순차적인 비교만 가능
2. **SQL Server**
   - TOP 절은 SELECT 문에서만 사용 가능
   - DELETE, UPDATE에서도 사용 가능
3. **MySQL**
   - LIMIT는 정상 위치에만 위치
   - 음수 값 사용 불가

6. 실전 활용 예제

6.1 페이징 처리

```
-- Oracle
SELECT *
FROM (
  SELECT a.*, ROWNUM as rnum
  FROM (
    SELECT *
    FROM 게시판
    ORDER BY 작성일 DESC
  ) a
  WHERE ROWNUM <= 20
) WHERE rnum > 10;

-- MySQL
SELECT *
FROM 게시판
ORDER BY 작성일 DESC
LIMIT 10, 10;  -- 11번째부터 10개
```

6.2 복합 조건 Top N

```
-- 부서별, 직급별 급여 상위 3명
SELECT *
FROM (
  SELECT 사원.*,
      ROW_NUMBER() OVER (
          PARTITION BY 부서번호, 직급
          ORDER BY 급여 DESC
      ) as rn
  FROM 사원
) WHERE rn <= 3;
```

SQLD 시험 대비 TIP

주요 출제 포인트

1. **DBMS별 구현 방식**
   - ROWNUM, TOP, LIMIT의 차이
   - 각 방식의 특징
2. **동률 처리의 방식**
   - ROW_NUMBER, RANK, DENSE_RANK의 차이
   - WITH TIES 활용
3. **그룹별 Top N**
   - PARTITION BY 활용
   - 복합 조건 처리

학습 전략

1. DBMS별 문법 차이 이해
2. 동률 처리 방식 숙지
3. 다양한 예제 실습
4. 성능 관련 내용 학습

실전 문제 유형

1. 결과 집합 예측
2. 적절한 구현 방식 선택
3. 성능 최적화 방안 도출
4. 복합 조건 처리 방법