WHERE 절의 목적과 사용법

SQL의 'WHERE' 절은 레코드를 필터링하고 쿼리 결과에 반환되는 행을 제한하는 데 사용됩니다. 이 절은 출력에 포함될 행에 대해 충족되어야 하는 조건을 지정합니다.

WHERE 절의 구문

SQL 문의 'WHERE' 절에 대한 기본 구문은 다음과 같습니다.

```
SELECT column1, column2, ...
FROM table_name WHERE condition;
```

- SELECT 는 검색할 열을 지정합니다.
- FROM 은 데이터를 검색할 테이블을 식별합니다.
- WHERE 는 충족해야 하는 조건을 정의합니다.

조건 유형

WHERE 절에 지정된 조건에는 다양한 연산자가 포함될 수 있습니다.

1 **비교 연산자**
=, >, <, >=, <=, <>(같지 않음)

2 **논리 연산자**
AND, OR, NOT

3 **IN**
한 열에 가능한 여러 값을 지정합니다.

4 **BETWEEN**
값 범위 지정

5 **LIKE 및 ILIKE**
패턴 일치를 위한

다른 SQL 명령과 함께 WHERE 사용

'WHERE' 절은 다음을 포함한 다양한 SQL 명령과 함께 사용할 수 있습니다.

SELECT
결과 집합의 레코드를 필터링합니다.

UPDATE
테이블에서 업데이트해야 할 행을 지정합니다.

DELETE
테이블에서 제거해야 하는 행을 나타냅니다.

WHERE 절 사용법의 예

1 **정확한 값으로 필터링**
WHERE age = 25

2 **범위별 필터링**
18세에서 30세 사이의 나이

3 **조건 결합**
WHERE 나이 > 20 AND 성별 = '여성'

4 **패턴 검색**
WHERE name LIKE 'Jo%'("Jo" 로 시작하는 모든 이름 검색)

성능 고려 사항

쿼리 성능을 향상하려면 'WHERE' 절을 효율적으로 사용하는 것이 중요합니다. 'WHERE' 절에 사용된 열을 기반으로 하는 적절한 인덱싱은 쿼리 실행 속도를 크게 높일 수 있습니다.

일반적인 함정

1 **잘못된 논리 조건**
너무 많거나 너무 적은 행을 검색할 수 있습니다.

2 **패턴 일치 오류**
패턴 일치에 LIKE 대신 =를 사용하면 올바른 결과가 반환되지 않습니다.

복잡한 조건

논리 연산자를 사용하여 여러 조건을 결합하여 더 복잡한 쿼리를 구성할 수 있습니다.

AND
결합된 모든 조건이 충족되어야 할 때 사용됩니다.

OR
조건 중 하나 이상이 충족되어야 할 때 사용됩니다.

NOT
조건을 부정하는 데 사용됩니다.

평가 순서를 제어하기 위해 괄호를 사용하여 복잡한 WHERE 절을 구성할 수 있습니다. 예를 들어:

```
WHERE (age >= 18 AND age <= 30) AND (status = 'Single' OR status = 'Divorced');
```

Null 값 처리

SQL에서 'NULL'은 누락되었거나 알 수 없는 데이터를 나타냅니다. WHERE 절에서 NULL 값을 처리하려면 특수 연산자가 필요합니다.

1 **IS NULL**
NULL 값을 확인합니다.

2 **IS NOT NULL**
NULL이 아닌 값을 확인합니다.

예를 들어: WHERE birthdate IS NULL;

JOIN 과 함께 WHERE 사용

'WHERE' 절은 조인 조건을 지정하거나 조인 결과를 필터링하기 위해 테이블 간의 조인과 관련된 쿼리에도 유용합니다. 예를 들어:

```
SELECT * FROM employees JOIN departments ON employees.department_id = departments.id
WHERE departments.name = 'Human Resources';
```

WHERE 절의 하위 쿼리

하위 쿼리는 다른 쿼리 내에 중첩된 쿼리입니다. 'WHERE' 절에 하위 쿼리를 사용하면 데이터베이스 자체에 포함된 데이터를 기반으로 쿼리를 더욱 동적으로 만들 수 있습니다. 예를 들어:

```
SELECT * FROM employees WHERE department_id IN (SELECT id FROM departments WHERE name = 'IT');
```

성능 팁

1 **선택성**
처리되는 행 수를 줄이므로 선택성이 높은 조건을 선호합니다.

2 **Sargability**
'WHERE' 절의 조건은 "Sargable" 이어야 합니다. 즉, 인덱스를 활용할 수 있어야 합니다. WHERE name = 'John' 대신 WHERE UPPER(name) = 'JOHN' 과 같이 인덱스 사용을 방해할 수 있는 열에 대한 함수나 작업을 피해야 합니다.

일반적인 오류

1 **구문 오류**
잘못된 연산자나 잘못된 논리 커넥터 배치로 인해 오류가 발생하거나 예상치 못한 결과가 발생할 수 있습니다.

2 **데이터 유형 불일치**
데이터 유형에 잘못된 연산자를 적용합니다(예: 날짜에 문자열 비교 연산자 사용).