정규 표현식 개념서

이 문서는 SQLD 시험을 위한 정규 표현식 개념서입니다. 정규 표현식의 기본 개념, 패턴 문자, DBMS별 구현 방식, 주요 패턴 예제, 고급 활용 예제, 성능 고려사항, 주의사항 및 제약사항, 그리고 SQLD 시험 대비 팁을 다룹니다. 데이터베이스에서 문자열 패턴을 효과적으로 다루는 방법을 상세히 설명합니다.

1. 정규 표현식의 기본 개념

1.1 정의

- 특정한 패턴을 가진 문자열을 표현하는 방법
- 문자열의 검색, 치환, 추출 등에 사용
- DBMS에서 REGEXP 또는 SIMILAR TO로 구현

1.2 주요 용도

1. **데이터 검증**
   - 이메일, 전화번호, 주민번호 등의 형식 검사
2. **문자열 검색**
   - 특정 패턴을 가진 데이터 추출
3. **문자열 치환**
   - 패턴에 맞는 문자열 변경

2. 기본 패턴 문자

2.1 문자 클래스

```
-- 기본 문자 클래스
[abc]    -- a, b, c 중 하나와 매치
[^abc]   -- a, b, c를 제외한 문자와 매치
[a-z]    -- a부터 z까지의 문자와 매치
[0-9]    -- 숫자와 매치
[a-zA-Z] -- 영문자와 매치
```

2.2 메타 문자

```
-- 주요 메타 문자
.    -- 임의의 한 문자
^    -- 문자열의 시작
$    -- 문자열의 끝
\    -- 특수문자를 일반문자로 취급
|    -- 또는(OR)
()   -- 그룹화
```

2.3 수량 한정자

```
-- 수량 지정
*    -- 0회 이상 반복
+    -- 1회 이상 반복
?    -- 0회 또는 1회
{n}  -- n회 반복
{n,} -- n회 이상 반복
{n,m} -- n회 이상 m회 이하 반복
```

3. DBMS별 정규 표현식 구현

3.1 Oracle

```
-- REGEXP_LIKE: 패턴 매칭 검사
SELECT *
FROM 고객
WHERE REGEXP_LIKE(전화번호, '^01[0-9]-[0-9]{3,4}-[0-9]{4}$');

-- REGEXP_SUBSTR: 패턴에 맞는 문자열 추출
SELECT REGEXP_SUBSTR(이메일, '[^@]+@[^@]+\.[^@]+')
FROM 고객;

-- REGEXP_REPLACE: 패턴에 맞는 문자열 치환
SELECT REGEXP_REPLACE(전화번호, '(02|031|032)-', '지역번호-')
FROM 고객;
```

3.2 MySQL

```
-- REGEXP: 패턴 매칭
SELECT *
FROM 고객
WHERE 전화번호 REGEXP '^01[0-9]-[0-9]{3,4}-[0-9]{4}$';

-- REGEXP_REPLACE: 문자열 치환
SELECT REGEXP_REPLACE(전화번호, '^(02|031|032)-', '지역번호-')
FROM 고객;
```

3.3 PostgreSQL

```
-- SIMILAR TO: 패턴 매칭
SELECT *
FROM 고객
WHERE 전화번호 SIMILAR TO '01[0-9]-[0-9]{3,4}-[0-9]{4}';

-- ~ 연산자: POSIX 정규 표현식
SELECT *
FROM 고객
WHERE 전화번호 ~ '^01[0-9]-[0-9]{3,4}-[0-9]{4}$';
```

4. 주요 패턴 예제

4.1 숫자 패턴

```
-- 숫자만 포함
^\d+$

-- 소수점 포함 숫자
^\d+(\.\d+)?$

-- 3자리마다 콤마가 있는 숫자
^\d{1,3}(,\d{3})*$
```

4.2 문자열 패턴

```
-- 영문자만 포함
^[a-zA-Z]+$

-- 한글만 포함
^[가-힣]+$

-- 영문과 숫자 조합
^[a-zA-Z0-9]+$
```

4.3 특수 형식

```
-- 이메일 주소
^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$

-- 전화번호
^01[0-9]-\d{3,4}-\d{4}$

-- IP 주소
^([0-9]{1,3}\.){3}[0-9]{1,3}$
```

5. 고급 활용 예제

5.1 데이터 검증

```
-- 유효한 이메일 주소 검색
SELECT 이메일
FROM 고객
WHERE REGEXP_LIKE(이메일,
    '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$');

-- 올바른 주민번호 형식 검사
SELECT 주민번호
FROM 고객
WHERE REGEXP_LIKE(주민번호,
    '^[0-9]{6}-[1-4][0-9]{6}$');
```

5.2 데이터 추출

```
-- 문자열에서 숫자만 추출
SELECT REGEXP_REPLACE(주소, '[^0-9]', '') AS 번지수
FROM 주소;

-- 도메인 추출
SELECT REGEXP_SUBSTR(이메일, '@[^@]+$') AS 도메인
FROM 이메일;
```

5.3 데이터 변환

```
-- 전화번호 형식 통일
SELECT REGEXP_REPLACE(
    전화번호,
    '(02|031|032)-([0-9]{3,4})-([0-9]{4})',
    '\1-\2-\3'
) AS 변환된_전화번호
FROM 고객;
```

6. 성능 고려사항

66.1 인덱스 활용

- 정규 표현식은 인덱스를 활용하지 못함
- 가능한 경우 LIKE 검색으로 대체
- 대용량 데이터 처리 시 주의

6.2 최적화 방안

1. **패턴 단순화**
   - 복잡한 패턴은 성능 저하의 원인
   - 가능한 단순한 패턴 사용
2. **데이터 필터링**
   - 정규식 검색 전 데이터 축소
   - WHERE절 활용 통한 사전 필터링

7. 주의사항 및 제약사항

7.1 DBMS별 차이점

1. **문법 차이**
   - DBMS별로 지원하는 함수명 상이
   - 정규식 문법 지원 범위 차이
2. **성능 차이**
   - DBMS별 정규식 처리 성능 차이
   - 최적화 방식의 차이

7.2 일반적인 제약사항

1. **복잡도 제한**
   - 너무 복잡한 패턴은 가독성 저하
   - 유지보수의 어려움
2. **성능 제약**
   - 대용량 데이터 처리 시 성능 저하
   - 인덱스 활용 불가

SQLD 시험 대비 TIP

주요 출제 포인트

1. **기본 패턴 문자**
   - 메타 문자의 의미
   - 문자 클래스 활용
2. **DBMS별 구현 방식**
   - 주요 정규식 함수
   - 문법 차이점
3. **패턴 매칭**
   - 주요 패턴의 이해
   - 결과 예측

학습 전략

1. 기본 패턴 문자 숙지
2. 자주 사용되는 패턴 학습
3. DBMS별 차이점 이해
4. 실제 예제 실습

실전 문제 유형

1. 패턴 매칭 결과 예측
2. 적절한 패턴 작성
3. 성능 최적화 방안
4. DBMS별 구현 방식