2023-03-02

SQL(MySQL) & NoSQL(MongoDB)

MySQL YUM Repository - 레드햇계열 패키지

MySQL APT Repository- 우분투계열 패키지

▼ Database 정의

한 조직의 여러 응용 시스템들이 공용(Shared)하기 위해 통합(Integrated)하고 저장(Stored)한 운영 데이터(Operational data)의 집합

▼ Database 특징

- 컴퓨터 시스템과 무관
- 데이터의 구조적 집합(종이,장부 등도 데이터베이스)
- 일반적으로 컴퓨터 시스템을 사용하여 구축한 데이터의 집합
- 데이터 모델에 따라 데이터베이스의 구조는 달라질 수 있다.

DBMS

DBMS Client Tools

- MySQL : Workbench
- TOAD, Orange, SQL Gate for oracle, SQL Developer

SQL 관계형 모델(RDB)

▼ Key Term

- 데이터베이스(Database) : Relation의 집합
- 릴레이션(Relation) : Tuple의 집합
- 튜플(Tuple) : 테이블의 Row
- 애트리뷰트(Attribute) : 릴레이션의 특징을 나타내는 단일한 데이터이며 컬럼
- 도메인(Domain) : 특정 Attribute가 가질 수 있는 값의 집합

```
# 커미션을 받는 직원 조회 commission_pct
SELECT * FROM employees WHERE commission_pct <> NULL;
# 위와 같이 쿼리문을 작성할 경우 원하는 결과값이 조회되지 않음
-------------------------------------------------------------
# commission_pct 컬럼에서 null이 아닌 경우로 조회하고 싶은 경우 IS NOT NULL 연산자 사용.
SELECT * FROM employees WHERE commission_pct IS NOT NULL;
```

NULL - 값이 없음을 의미하나 **정해지지 않은 값**이라는 형태로 인식할 필요

▼ SQL - SELECT 명령어 Where절

- 연산자
  - 산술, 비교, 논리 연산자 사용
    - IN
    - BETWEEN a AND/OR b
    - LIKE
    - IS NULL, IS NOT NULL
    - AND
    - OR

        - NOT
        - ANY, ALL : 집합 중 어느 하나 혹은 집합 중 모든 열
        - EXISTS : 존재 유무에 따라(subquery에서 사용)
    - 연산자 우선 순위 (산술 → 비교 → 논리)
        - Arithmetic operator
        - Concatenation operator
        - Comparisono condition
        - IS[NOT] NULL, LIKE, [NOT]IN
        - [NOT] BETWEEN
        - NOT logical condition
        - AND logical condition
        - OR logical condition

▼ SQL - 집계함수

- AVG () - 평균
- MAX() - 최대값
- MIN() - 최소값
- LAG() - N번째 이전 값
- LEAD() - N번째 이후 값
- PERCENT_RANK() - 퍼센트 랭킹값
- RANK() - 랭킹값
- ROW_NUMBER() - 레코드 순번
- DENSE_RANK() - 동점 랭킹은 하나의 랭킹으로 계산하여 구한 랭킹값
- SUM(col) - NULL제외 해당 Row 값 총합
- COUNT(col) - NULL제외 해당 Row 값 개수
- COUNT (*) - 해당 Row의 총 개수

NULL인 데이터는 집합함수에 참여하지 않음.

SQL 인덱스

▼ 인덱스 개념

- 테이블의 컬럼에 대응되는 별도의 객체로 독립적인 저장공간 보유
- Optimizer가 최적의 실행경로를 설정하는데 중요한 Factor역할
- 인덱스를 생성시킨 컬럼의 값(VALUE)과 Row의 물리적인 주소(ctid)로 생성
- 성능 향상 :
    - 별도의 정렬없이 결과 추출가능
    - 물리적인 디스크 I/O 감소

▼ 참고사항

- 기존과 동일한 질의문 작성
- 조회하고자 하는 자료에 보다 빠르게 접근하는 수단
- 활용원리를 알고 어떻게 설계하는지가 중요함
- 테이블에는 저장된 데이터가 차지하는 공간 이외에도 별도의 저장공간이 필요
- 만일 테이블에 저장된 데이터를 자주 변경해야할 경우 관련된 Index역시 함께 수정해야하기 때문에 오버로드로 인한 성능저하 가능

인덱스는 쿼리의 성능에 도움을 준다.

인덱스 선정 손익분기점 : **10~15%**

SINGLE BLOCK I/O의 경우 10~15% **이하** = 효율적

MULTI BLOCK I/O의 경우 10~15% **이상** = 효율적

SQL JOIN

▼ Cross Join

- 테이블 Row의 모든 조합 조회(Cartesian Product)

▼ Inner Join

- Join 조건을 만족하는 튜플만 조회

▼ Outer Join

- Join 조건을 만족하지 않는 튜플(짝이 없는 튜플)도 NULL과 함께 조회
  - Left Outer Join
    - 왼쪽 테이블의 모든 튜플중 Join조건을 만족하지 못한 NULL값도 함께 조회
  - Right Outer Join
    - 오른쪽 테이블의 모든 튜플 조회. NULL값도 함께 조회
  - Full Outer Join
    - 양쪽 테이블 내부의 데이터 중 Join 조건을 만족시키지 못한 모든 Row 조회

▼ Non Equi-Join

- =이 아닌 조건에 의한 조인 결과 조회

▼ Self Join

- 자기 자신과 조인한 결과 조회

▼ Natural Join

- Join하려는 테이블들의 공통 컬럼을 기준으로 Join. Inner Join과 유사하나 중복컬럼이 없는것이 특징

SQL 서브쿼리

쿼리 내부에 포함되어있는 쿼리

▼ 위치에 따른 분류

- 인라인 뷰(Inline View) : From절
- Nested 서브쿼리 : Where절
- Scalar 서브쿼리 : Select절

▼ 동작 방식에 따른 분류

- NESTED Subquery
- 상호연관(Correlated) Subquery

▼ 리턴되는 데이터에 따른 분류

- 단일컬럼(Single Column) Subquery
- 다중컬럼(Multi Column) Subquery
- 단일행(Single Row) Subquery
- 다중행(Multi Row) Subquery

NoSQL 비 관계형 모델

NoSQL은 “SQL이 아니다”라는 의미이나 Not Only SQL이라는 의미를 부여하기도 한다.

Document - 튜플(sql)

Collection - 테이블(sql)

▼ 과제 1

성명, 이메일주소, 연봉, 입사후30주년일자, 근속년수(소수2자리까지), 입사일자를 출력하시오.

- 성명은 붙여서 출력합니다.
- 이메일주소는 @kosa.com을 붙입니다.
- 연봉은 커미션을 반영하고, 급여의 12배, 환율은 1240으로 하며, 천단위로 버림합니다.
- 입사후 30주년기념일자는 '일-월-년'으로 출력합니다.
- 80번부서로 한정합니다.
- 근속년수로 많은 순서로 정렬합니다.
- optional) 30주년기념식일자도 출력합니다. 기념식일자는 30주년이 지난 다음주 월요일입니다.

```
SELECT CONCAT(first_name, last_name)성명,
CONCAT(email, '@kosa.com')이메일주소,
TRUNCATE((salary*(1 +  IFNULL(commission_pct,0))*12)*1240, -3) 연봉,
DATE_FORMAT(DATE_ADD(hire_date, INTERVAL 30 YEAR),'%d-%m-%Y') 입사30주년일자,
FORMAT(YEAR(NOW())-LEFT(hire_date,4),2)근속년수,
(hire_date) 입사일자
FROM employees WHERE department_id = 80
ORDER BY hire_date;
```

▼ 과제 2

1. 부서번호, 부서명, 부서장사번, 부서장성명, 부서장입사일자를 출력하시오.
   - 단 부서장이 없으면 부서장없음으로 출력
2. 자신의 관리자보다 먼저 입사한 직원의 사번, 성명, 입사일자, 관리자입사일자를 출력하시오.
3. Seattle에 근무하는 직원의 사번, 성명, 입사일자, 부서번호, 부서명을 출력
   - 단 서브쿼리로 합니다.

```
# 1
SELECT d.department_id, d.department_name, IFNULL(d.manager_id, '부서장없음')manager_id, e.first_name, e.last_name, e.hire_date
FROM departments d LEFT OUTER JOIN employees e ON d.manager_id = e.manager_id;

# 2
SELECT e.employee_id 사번, CONCAT(e.first_name, e.last_name)성명, e.hire_date 입사일자, m.hire_date 관리자입사일자
FROM employees e, employees m WHERE e.manager_id = m.employee_id AND e.hire_date < m.hire_date;

# 3
SELECT (e.employee_id)사번, CONCAT(e.first_name, e.last_name)성명, (e.hire_date)입사일자, (e.department_id)부서번호, (d.department_name)
FROM employees e, departments d WHERE e.department_id = d.department_id AND d.location_id = (SELECT l.location_id
                                            FROM locations l
                                            WHERE l.city = 'Seattle');
```

강사님 코드

```
# 1
SELECT d.department_id, d.department_name, IFNULL(e.employee_id, '부서장없음')manager_id, e.first_name, e.last_name, e.hire_date
 FROM employees e JOIN departments d ON e.employee_id = d.manager_id;
# 2
SELECT e.employee_id, e.last_name, e.first_name, e.hire_date, m.last_name, m.hire_date
FROM employees e JOIN employees m
         ON e.manager_id = m.employee_id WHERE e.hire_date < m.hire_date;
# 3
SELECT e.emplouee_id, e.last_name, e.hire_date, e.department_id, (SELECT department_name FROM departments
WHERE department_id = e.department_id) dept_name FROM employees e
WHERE department_id in (SELECT department_id FROM departments
WHERE location_id in (SELECT location_id FROM locations WHERE city='Seattle'));
```