**고급 데이터베이스**

**[HW3] IndexConcept**

|  |  |
| --- | --- |
| 학번 / 이름 | 안상혁  쑨뤠이  신예림  신정웅  신호진  32131766 황준일 |
| 담당교수 | 이 석 균 교수님 |
| 제출일 | 2019. 10. 17 (목) |

1. Employees Database 생성 ( indexed )

|  |
| --- |
| # 다음 쿼리문으로 데이터가 정상적을 들어갔는지 확인할 수 있다.  SELECT  (SELECT count(\*) as cnt from employees.departments) as departments\_cnt,  (SELECT count(\*) as cnt from employees.dept\_emp) as dept\_emp\_cnt,  (SELECT count(\*) as cnt from employees.dept\_manager) as dept\_manager\_cnt,  (SELECT count(\*) as cnt from employees.employees) as employees\_cnt,  (SELECT count(\*) as cnt from employees.salaries) as salaries\_cnt,  (SELECT count(\*) as cnt from employees.titles) as titles\_cnt;  # 결과 확인    합쳐서 3919015 개의 데이터가 삽입 되었음을 확인할 수 있다. |

1. employees2 Database 생성 ( non-indexed )

|  |
| --- |
| # 인덱스 제거 후 Table 생성  CREATE TABLE employees (  emp\_no INT NOT NULL,  birth\_date DATE NOT NULL,  first\_name VARCHAR(14) NOT NULL,  last\_name VARCHAR(16) NOT NULL,  gender ENUM ('M','F') NOT NULL,  hire\_date DATE NOT NULL  );  CREATE TABLE departments (  dept\_no CHAR(4) NOT NULL,  dept\_name VARCHAR(40) NOT NULL  );  CREATE TABLE dept\_manager (  dept\_no CHAR(4) NOT NULL,  emp\_no INT NOT NULL,  from\_date DATE NOT NULL,  to\_date DATE NOT NULL  );  CREATE TABLE dept\_emp (  emp\_no INT NOT NULL,  dept\_no CHAR(4) NOT NULL,  from\_date DATE NOT NULL,  to\_date DATE NOT NULL  );  CREATE TABLE titles (  emp\_no INT NOT NULL,  title VARCHAR(50) NOT NULL,  from\_date DATE NOT NULL,  to\_date DATE  );  CREATE TABLE salaries (  emp\_no INT NOT NULL,  salary INT NOT NULL,  from\_date DATE NOT NULL,  to\_date DATE NOT NULL  );  # 데이터 삽입 후 employees와 employees2의 결과 확인 SELECT  (SELECT count(\*) from employees.departments) as d\_cnt1,  (SELECT count(\*) from employees2.departments) as d\_cnt2,  (SELECT count(\*) from employees.dept\_emp) as de\_cnt1,  (SELECT count(\*) from employees2.dept\_emp) as de\_cnt2,  (SELECT count(\*) from employees.dept\_manager) as dm\_cnt1,  (SELECT count(\*) from employees2.dept\_manager) as dm\_cnt2,  (SELECT count(\*) from employees.employees) as e\_cnt1,  (SELECT count(\*) from employees2.employees) as e\_cnt2,  (SELECT count(\*) from employees.salaries) as s\_cnt1,  (SELECT count(\*) from employees2.salaries) as s\_cnt2,  (SELECT count(\*) from employees.titles) as t\_cnt1,  (SELECT count(\*) from employees2.titles) as t\_cnt2; |

# 질의문 생성

## 인덱스 사용 의도에 맞게 질의문 생성

### Equality

|  |  |  |
| --- | --- | --- |
| **SELECT \* FROM dept\_emp WHERE emp\_no = 100000;** | | |
| Indexed | |  |
| non-indexed | |  |

### 

### Range

|  |  |
| --- | --- |
| **SELECT \* FROM dept\_emp WHERE dept\_no < 'd002';** | |
| indexed |  |
| non-indexed |  |

### Group

|  |  |
| --- | --- |
| **SELECT emp\_no, count(salary) AS salary\_cnt FROM salaries GROUP BY emp\_no;** | |
| Indexed |  |
| non-indexed |  |

### 

### Order By

|  |  |
| --- | --- |
| **SELECT \* FROM employees ORDER BY emp\_no ASC;** | |
| Indexed |  |
| non-indexed |  |

## 

## 인덱스 사용 의도에 맞지 않게 질의문 생성

### Non-indexd field scan

|  |  |
| --- | --- |
| **SELECT \* FROM employees WHERE first\_name = 'George';** | |
| Indexed |  |
| non-indexed |  |

## 

### Bad range scan

|  |  |
| --- | --- |
| **SELECT \* FROM dept\_emp WHERE emp\_no < 30000 or emp\_no > 200000** | |
| Indexed |  |
| non-indexed |  |

### 

### Not equlity

|  |  |
| --- | --- |
| **SELECT \* FROM dept\_emp WHERE emp\_no != 100000;** | |
| Indexed |  |
| non-indexed |  |

## 

## index가 주는 영향 분석

## index를 지정한 field(column)에 대해 조건 검색을 수행할 경우, single scan, range scan 등을 수행하기 때문에 수행 시간이 빠르다.

## index를 지정하지 않을 경우 조건에 상관 없이 항상 full table scan을 실행한다.

## index를 지정하지 않은 field(column)에 대해 조건 검색을 수행할 경우, 수행 시간에는 큰 차이가 없다.

## index를 사용하면 select 명령은 빠르게 수행하지만, insert, update, delete 등에서는 성능저하가 발생한다. 왜냐하면 index의 구조를 유지해야 하기 때문이다.

## 인덱스를 사용하더라도 적절하지 못한 조건을 사용할 경우, 오히려 더 큰 비용의 cost가 발생한다.

## index로 지정된 field는 정렬되기 때문에 order by 명령에 효과적이다.

## index로 지정된 field에 대해 group by를 수행할 경우 효과적이다.

# 

# JOIN을 사용하는 질의문 생성

## 비교해보기

|  |  |
| --- | --- |
| **SELECT** e.first\_name, e.last\_name, s.salary, t.title, de.from\_date **AS** de\_from, de.to\_date **AS** de\_to, dm.from\_date **AS** dm\_from, dm.to\_date **AS** dm\_to**FROM** employees e**JOIN** salaries s **ON** s.emp\_no = e.emp\_no**JOIN** titles t **ON** s.emp\_no = t.emp\_no**JOIN** dept\_emp de **ON** s.emp\_no = de.emp\_no**JOIN** dept\_manager dm **ON** s.emp\_no = de.emp\_no; | |
| Indexed | |
| non-indexed   * 성능 측정 불가능 * 10분이상 수행해도 결과가 나오지 않았음 | |

## 

## 성능차이가 발생하는 이유

## index가 없는 필드를 기준으로 JOIN을 하게 되면, Full Table Scan을 하면서 각 row마다 또 Full Table Scan을 하게 된다. 즉, 10000개의 튜플이 있는 테이블과 1000개의 튜플이 있는 테이블을 JOIN하면 10000 \* 1000 번의 scan을 수행한다.

## 반대로 index가 있는 필드를 기준으로 JOIN을 하게 되면 최소 1 row scan이 가능하다

## 5중첩 JOIN에 대한 질의문을 수행했을 때 index가 있는 경우에는 108.87 \* 1000000 번의 scan을 수행하고, index가 없는 경우에는 24 \* 299389 \* 331008 \* 442123 \* 2707962 번의 scan을 수행한다. (수행하다가 컴퓨터 망가질 뻔했다)

## 즉, index를 하지 않으면 무조건 full table scan \* full table scan 형태로 테이블을 조회한다.

## 이것은 데이터가 많아질수록 매우 치명적이다.

# employees2 개선

## Equality

|  |  |  |
| --- | --- | --- |
| **# 단일 Equlity만 사용하는 경우, 해당 필드만 Index로 만드는 게 좋다. ALTER TABLE `dept\_emp` ADD INDEX `index1` USING BTREE (`emp\_no`) VISIBLE;****SELECT \* FROM dept\_emp WHERE emp\_no = 100000;** | | |
| employees1 (기존) | | employees2 (개선) |

### 

## Range

|  |  |  |
| --- | --- | --- |
| **# 기존의 탐색 우선순위는 emp\_no > dept\_no 였다. 즉, emp\_no를 기준으로 정렬된 상태이다.****# 이것을 dept\_no > emp\_no로 변경시키면 dept\_no를 기준으로 정렬된 상태이기 때문에 성능이 더 좋아진다. ALTER TABLE `employees2`.`dept\_emp` ADD PRIMARY KEY (`dept\_no`, `emp\_no`);****SELECT \* FROM dept\_emp WHERE dept\_no < 'd002';** | | |
| employees1 (기존) | | employees2 (개선) |

# 인덱스의 효용성

## Clustered Index에 대한 적합한 예 clustered index에 대한 효과를 확실하게 확인할 수 있는 부분은 JOIN 질의문이다. 이러한 질의문을 사용할 경우, 인덱스가 없는 것과 있는 것은 하늘과 땅 차이임을 확인할 수 있다.

|  |  |
| --- | --- |
| **SELECT** e.first\_name, e.last\_name, s.salary, t.title, de.from\_date **AS** de\_from, de.to\_date **AS** de\_to, dm.from\_date **AS** dm\_from, dm.to\_date **AS** dm\_to**FROM** employees e**JOIN** salaries s **ON** s.emp\_no = e.emp\_no**JOIN** titles t **ON** s.emp\_no = t.emp\_no**JOIN** dept\_emp de **ON** s.emp\_no = de.emp\_no**JOIN** dept\_manager dm **ON** s.emp\_no = de.emp\_no; | |
| Indexed | non-indexed    이런 쿼리문 돌리면 컴퓨터 망가진다.. |

## 

## Index-only plan의 적합한 예 \* explain에서 type이 index일 경우 index only plan을 사용한다.

* + 1. 집계함수 사용(Group)

|  |
| --- |
| **SELECT emp\_no, count(\*) FROM titles WHERE FROM\_date > '1996-11-27' GROUP BY emp\_no;** |
|  |
| **SELECT emp\_no, count(salary) AS salary\_cnt FROM salaries GROUP BY emp\_no;** |
|  |

* + 1. 정렬 사용 (Order By)

|  |
| --- |
| **SELECT \* FROM employees ORDER BY emp\_no ASC;****SELECT \* FROM employees ORDER BY emp\_no DESC;** |
|  |