데이터베이스(다반)

20192875

조성현

중간고사 대체과제

**1. 트랜잭션의 4대 특징**

1. 원자성(Atomicity)

- 트랜잭션의 작업이 부분적으로 실행되거나 중단되지 않는 것

- 하나의 덩어리로 전부 되는 것 아니면 아무것도 되지 않음.

- 작업의 단위를 일부분만 실행하지 않음

1. 일관성(Consistency)

- 트랜잭션이 성공적으로 완료되면 일관적인 DB상태를 유지하는 것

ex) employees 테이블의 first\_name의 데이터 타입이 Char에서 int형으로 바뀌지 않음

- 트랜잭션이 진행되는 동안에 DB가 변경되더라도 업데이트된 DB로 트랜잭션

이 되는 것이 아니라 처음에 트랜잭션을 진행하기 위해 참조한 DB로 진행됨.

1. 격리성(Isolation)

- 트랜잭션 수행 시 다른 트랜잭션의 작업이 끼어들지 못하는 것

- 트랜잭션끼리 서로 간섭할 수 없음

1. 지속성(Durability)

- 성공적으로 수행된 트랜잭션은 영원히 반영이 되는 것

- 한번 commit을 하면 현재 상태는 영원히 보장됨

DB에서 일관성의 특징으로 사용자가 DB로 검색을 하는 등 이용할 때 서로 다른 사용자들이 일관성 있는 DB를 볼 수 있어서 서로 다른 DB를 보는 경우가 없다. 또한 일관성의 특징으로 employees 테이블에서 특정 사람이 부서를 옮겼을 때 개발 부서 테이블에서는 마케팅 부서로 옮겼다고 뜨는데 마케팅 부서에서는 아무 들어온 사람이 없는 경우가 없어야 한다. 일관성 특징으로 은행 DB에서 입출금 오류 등 심각한 문제를 막을 수 있어서 중요한 특징이다.

격리성은 한 DB에서 병행처리를 막아 공통된 데이터가 혼란스러워지는 경우를 막는 특징이다. 이와 같이 간섭이 일어날 경우 갱신분실, 오손판독, 반복불가능, 팬텀문제 등 여러가지 문제점들이 발생하기 때문에 Lock과 unlock 기능으로 격리성을 보장한다. 이 격리성은 DB를 이용하는데 서로 같은 데이터를 봐야하는 일관성 특징으로 비슷한 이유로 주요한 특징이다.

**2. MySQL의 인덱스 종류**

인덱스의 데이터 저장 방식별로 구분하는 것은 많은 분류가 가능하겠지만, 대표적으로 BTREE 인덱스와 HASH인덱스로 구분할 수 있다.

- BTREE

가장 일반적으로 사용되는 인덱스 알고리즘으로서, 칼럼의 값을 변형하지 않고, 원래의 값을 이용해 인덱싱하는 알고리즘

- HASH

컬럼의 값으로 해시 값을 계산해서 인덱싱하는 알고리즘으로, 매우 빠른 검색을 지원한다. 하지만 전방일치와 같이 값의 일부만 검색할 수 없고 비교문을 쓸 수 없다.

여기서 스토리지엔진에 따라 HASH를 못 쓸 수 있다.

MySQL 문법에서 테이블 생성 시 지정할 수 있는 인덱스 타입이 2개로 나타난다.

|  |
| --- |
| index\_type:  USING {BTREE | HASH} |

이는 문법 호환성을 위해서 문법적으로 허용만 할 뿐, MySQL은 스토리지엔진에 따라 지원되는 인덱스의 타입이 다르다.

|  |  |  |  |
| --- | --- | --- | --- |
| 스토리지엔진 | 지원되는 인덱스 | Full-text search indexes | Clustered 인덱스 |
| InnoDB | BTREE | No | Yes |
| Memorry/HEAP | HASH, BTREE | No | No |
| NDB | HASH, BTREE | No | No |
| MyISAM | BTREE, RTREE | Yes | No |
| Archive |  | No | No |

Clustered Index

테이블 레코드들이, 인덱스 컬럼의 정렬 순서대로 적재되어 있는 것. 인덱스 자체가 테이블 레코드 순서와 같다. 인덱스 자체의 리프 페이지가 곧 데이터이다. 그러므로 인덱스 자체에 데이터가 포함되어있다고 볼 수 있다.

MySQL에서 InnoDB를 스토리지엔진 기반에서의 경우

테이블에서 Primary key가 있다면 Primary key를 Clustered 인덱스로, 없다면 UNIQUE 하면서 NOT NULL인 컬럼을 Clustered 인덱스로, 그것도 없다면 임의로 보이지 않는 컬럼을 만들어 Clustered 인덱스로 만든다.

Secondary Index

보조 인덱스의 생성시에는 데이터 페이지는 그냥 둔 상태에서 별도의 페이지에 인덱스를 구성한다. 보조 인덱스의 인덱스 자체의 리프 페이지는 데이터가 아니라 데이터가 위치하는 주소값을 가진다.

Clustered 인덱스는 Secondary 인덱스 보다 검색(R) 속도는 더 빠르다. 하지만 입력/수정/삭제(CUD)는 느리다.

**3. 와일드카드(like)기반 검색**

검색하려는 값을 정확하게 모를 경우에도 검색할 수 있도록 하는 것

|  |  |
| --- | --- |
| 와일드 카드 | 기술 |
| % | 0개 이상의 문자를 대체 |
| \_ | 단일 문자에 대한 대체 |
| [charlist] | 문자 세트 및 범위가 일치 |
| [^charlist] 또는 [!charlist] | 문자 세트를 제외한 문자들로 대체 |

-예시

1) A%

‘A’로 시작하는 모든 문자열

2) ‘%A’%

‘A’가 포함된 모든 문자열

3) ‘\_A%’

두 번째 문자가 ‘A’인 모든 문자열

4) ‘[ABC]%’

첫 번째 문자가 ‘A’ 또는 ‘B’ 또는 ‘C’인 모든 문자열

5) ‘[^A]%’

첫 번째 문자가 ‘A’가 아닌 모든 문자열

6)[0-9][0-9][0-9][0-9]

4자리 숫자를 가지는 문자열

-단점

와일드카드 문자를 이용한 검색은 텍스트 열(문자열)에서만 사용할 수 있다.

NULL값은 와일드 카드로 매칭되지 않음

‘%’나 ‘\_’ 또는 ‘[‘ 등 와일드카드 연산자를 검색할 때는 주의가 필요함

참고문헌 및 링크

<https://victorydntmd.tistory.com/129>

<https://mommoo.tistory.com/62>

<http://jason-heo.github.io/mysql/2014/03/05/char13-index-types.html>

<https://velog.io/@ansrjsdn/MySQL-INDEX-%EB%9E%80>

<https://ggmouse.tistory.com/131>

<https://www.w3bai.com/ko/sql/sql_wildcards.html>

<https://velog.io/@datata29/SQL-%EC%99%80%EC%9D%BC%EB%93%9C-%EC%B9%B4%EB%93%9C-%EB%AC%B8%EC%9E%90%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%9C-%ED%95%84%ED%84%B0%EB%A7%81>