관계형 데이터베이스

Date	@2021/09/10

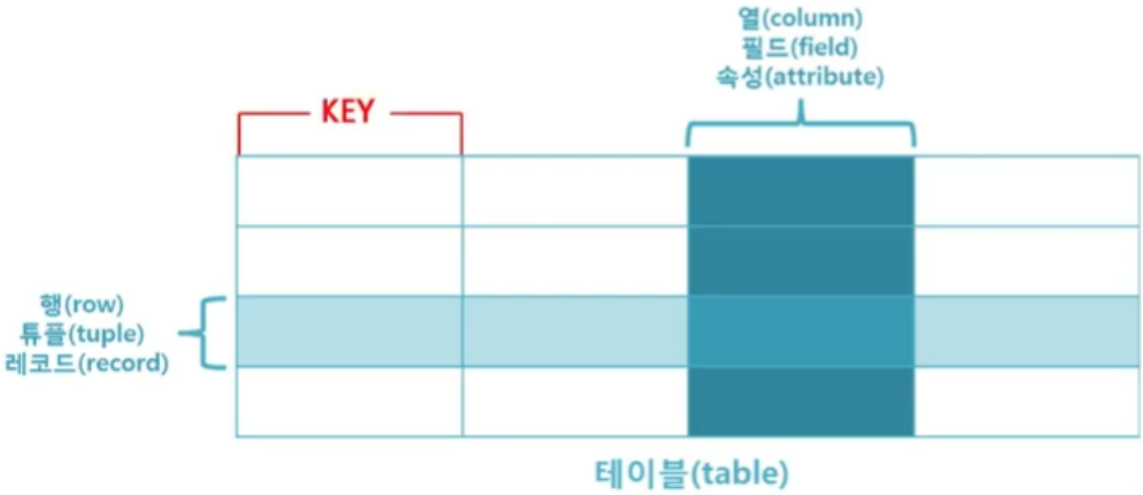
데이터베이스

- 데이터를 모아두는 곳
- 사람들이 원하는 자료를 쉽게 찾을 수 있다.
- 많은 데이터가 있어도 빠르게 찾을 수 있다.

관계형 데이터베이스

- 데이터를 Key, Value 형태로 식별이 가능하도록 함 ⇒ Column, Row
- 테이블 간에 데이터가 1:1, 1:N, N;N 형태로 관계가 존재한다
- 각 테이블 간의 Key를 통해 서로를 연결한다.

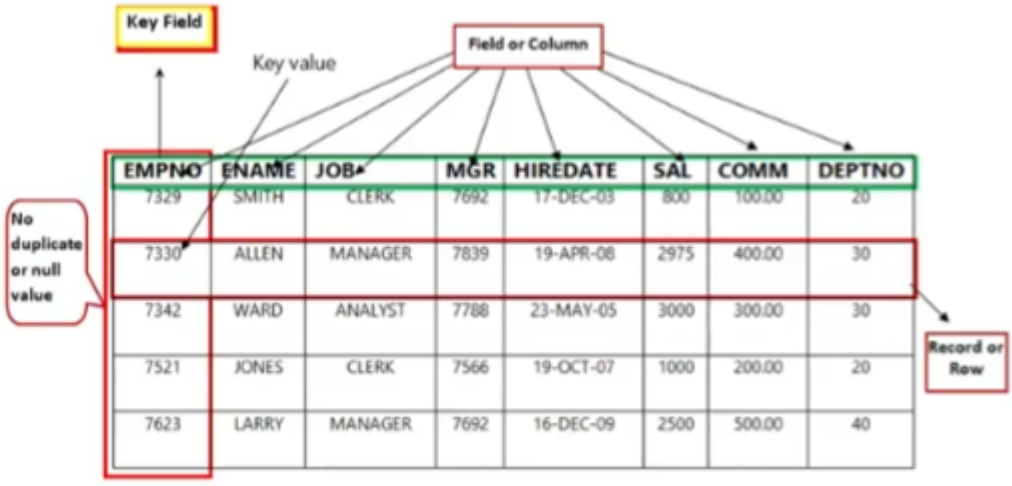
테이블 기본 구조



Primary Key

- 다른 항목과 중복되어 나타날 수 없는 단일 값(Unique)
- 다중 컬럼을 Primary Key로 설정할 수 있다.
- 관계형(Relation)을 완성할 수 있는 가장 중요한 정보

Key



PK와 INDEX

1. PK를 생성/삭제하면 INDEX 생성/삭제됨

2. INDEX : 데이터베이스 테이블의 검색 속도를 향상시키기 위한 자료구조
    a. 테이블을 대한 동작의 속도를 높여주는 자료 구조
    b. Indexes Table이라고 한다. 즉 Index도 테이블의 한 종류이다. (데이터베이스 객체의 한 종류)
    c. PK가 아닌 컬럼이라도 INDEX를 생성할 수 있다.

데이터베이스 객체

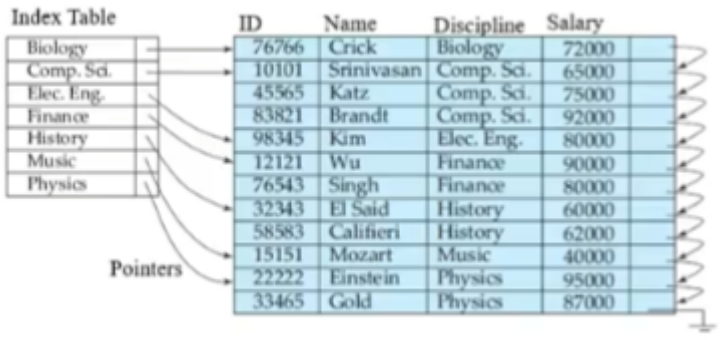
- 데이터베이스 내에 존재하는 논리적인 저장 구조
- Table, View, Indexes, synonym, sequence 등
- DDL(Data Defince Language)로 생성, 수정 및 삭제가 가능하다.

DDL

- SQL(Structured Query Language)의 종류 : 데이터 베이스에서 사용되는 언어
- DML(Data Manipulation Language) : SELECT, INSERT, UPDATE, DELETE
- DDL(Date Define Language) : CREATE, DROP 등 객체 관리
- DCL(Data Control Language) GRANT, REVOKE. 객체에 대한 접근 권한 관리
- TCL(Transaction Control Language) : COMMIT, ROLLBACK 등 DML 조작 결과를 제어

INDEX

INDEX의 동작



Pointer, Hash, B+ Tree의 알고리즘을 활용해서 더욱 편하게 찾고자한다.

INDEX를 왜 쓸까?

Q. 테이블 데이터가 100만건이면 오래걸리지 않을까?

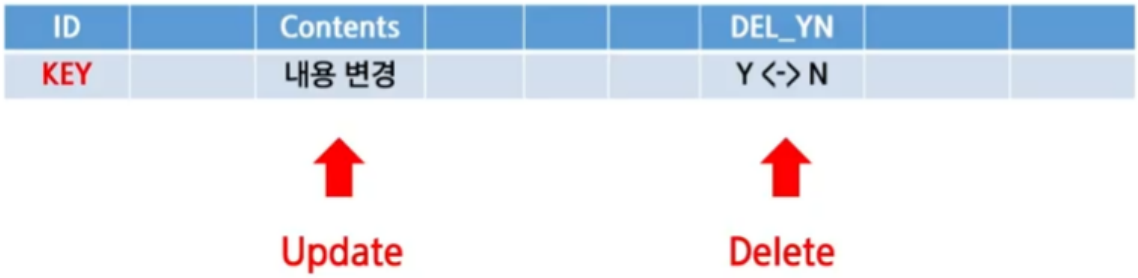
A. Indexex란 DB 자료구조를 통해 빠르게 조회할 수 있다.

Q. 인덱스를 많이 만들면 되겠다!

A. 인덱스를 잘못 관리하면 오히려 성능이 저하될 수 있다.

INDEX를 활용하면 안되는 경우

- DELETE, UPDATE가 자주 일어나는 경우



INDEX의 장점

- 테이블을 조회하는 속도&성능 향상
- 시스템의 부하를 줄일 수 있다
- 원하는 데이터를 빠르게 찾을 수 있다.

INDEX의 단점

- DB 저장 공간을 차지한다. (꽤 큰 용량을 잡아먹는다.)
- 인덱스를 관리하기 위한 작업이 필요하다.
- 잘못 사용할 경우 성능이 저하될 수 있다. (적재적소에 활용이 필요함)

INDEX를 잘 쓰려면?

- Where 조건 작성시 Index를 적용할 수 있는 컬럼 조건을 상단에 작성한다.

- Index로 사용된 컬럼은 컬럼값 그대로 사용한다.

  적용 O : Where Age > 18

  적용 X : Where Age * 2 > 36

- Between, like, <, > 등 범위 조건 뒤에 Index들은 적용되지 않으니 가장 마지막에 활용한다.

  적용 O : Where Id='8' and Age > 16

  적용 X : Where Age > 16 and Id = '8' → Age > 16까지는 Index가 적용되지만 뒤는 적용되지X

쿼리 성능을 향상시키려면?

- Table 생성시 PK, Index 등을 잘 구축한다.
- 같은 결과를 출력하더라도 최적화된 쿼리를 작성해야 한다.
- Table Join 순서도 중요하다. (Driving, Driven 테이블)
- 키워드 : Database Optimizer, Query Tuning, Query Plan