**20176342 송민준**

관계 DB 저장 시스템 개발 설계 중간 제출

**구현 언어 및 개발 환경**

Python 3.10.x

Visual Studio Code

**메타 파일 설계**

메타 파일은 테이블명.meta 파일로 각 테이블명 디렉토리에 저장되며,

메타 파일의 내용은 아래와 같이 테이블명, 생성일자, 컬럼과 컬럼의 형태(CHAR, VARCHAR), 그리고 해당 디렉토리에서 사용하는 슬롯들의 파일명(슬롯단위 파일I/O)이 저장되어 있고,

총 슬롯 페이지 수와 레코드 수가 저장되어 있습니다.

|  |
| --- |
| **Ex) student 테이블 meta 파일 (student.meta)**  파일 내용물  student.meta [2022-05-06 00:00:01] (생성일자 기록)  id CHAR(10)  name VARCHAR(20)  grade CHAR(3)  **0001.slot**  **0002.slot**  **0003.slot**  3 SLOTS (총 슬롯 페이지 수)  120 ROWS (총 레코드 수) |

**Meta-data의 Meta-data 설계**

저장시스템 home directory에 meta.meta 파일 생성 후 각 table.meta 파일의 위치와 개수 등의 정보를 가지게 설계합니다.

아래와 같이 3개의 테이블이 있는 DBMS 시스템에서는 3개의 메타파일이 생성되고 각각의 디렉토리 경로가 meta.meta 파일에 저장됩니다.

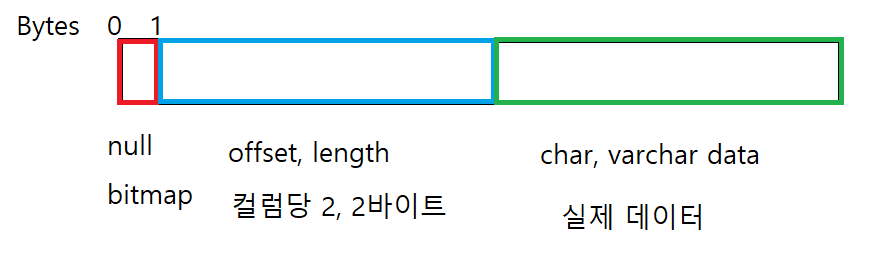
|  |
| --- |
| **Metafile [2022-05-06 00:00:01] 생성일자 기록**  student.meta ./student/ (메타 파일명, 폴더 경로)  player.meta ./player/  university.meta ./university/  3 meta files (메타 파일 개수) |

**파일 구조 설계**

저장시스템 home directory 기준으로, 테이블당 테이블명으로 폴더 생성. 각 폴더에는 slotted page의 개수 등 슬롯 페이지 관련 데이터를 담고 있는 table.meta 파일 존재.

각 슬롯 페이지 헤더에서는 레코드가 어디 있는지에 대한 Pointer를 담고 있게 설계합니다.

**가변길이 레코드 포맷**



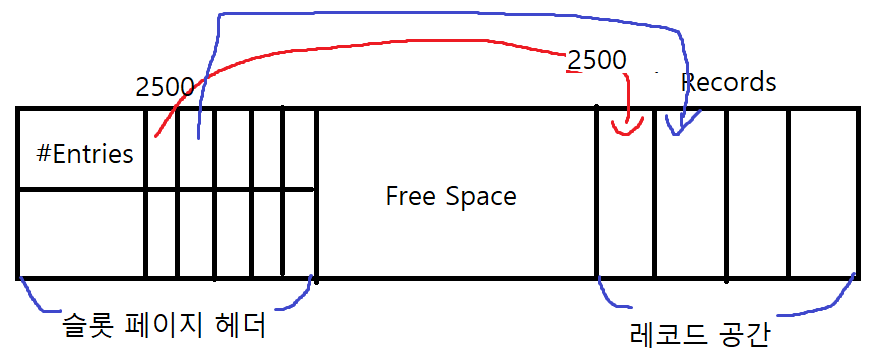
Null bitmap을 가변길이 레코드 제일 앞으로 오게 설계합니다.

Null bitmap은 8비트(1바이트)로 한 레코드에서 최대 컬럼을 8개만 가질 수 있게 설계합니다. Offset(2bytes) , Length(2bytes) 로 설계해서 총 Length(99)\*컬럼 개수(8) 대략 레코드 하나의 크기는 1000바이트가 최대가 되게 합니다. 한 슬롯 페이지는 4000바이트가 최대가 되게 구현을 할 것이므로 슬롯 하나에 레코드가 아무리 적게 들어가도 4개의 레코드가 들어가게 됩니다.

최소는 컬럼 1개\*length(1) + null bitmap 1바이트 => 3바이트가 되게 합니다. 이 경우에는 슬롯 하나에 레코드가 대략 1333개 정도 들어갈 수 있습니다. (4000/3=1333.333…)

Null bitmap이 1이 될 시 해당 컬럼 Byte 데이터 회수 설계합니다.

**Slotted page 구조**



슬롯 페이지 크기는 한 슬롯당 4000바이트로 설계합니다.

Free space 구현은 교재와 같이 슬롯 페이지의 중간 부분이 Free Space가 될 수 있도록 설계, #Entries 부분은 1바이트로 할당하여 슬롯 페이지 1개당 2^8-1개의 레코드를 최대로 가질 수 있게 설계,

레코드 포인터, End of Free Space 포인터는 2바이트씩 할당하여 4000바이트의 블록 크기를 처음부터 끝까지 가리킬 수 있게 설계,

레코드 포인터가 2바이트인 이유는 1바이트로는 2^8=64 바이트 까지만 가리킬 수 있기 때문에

2바이트 = 2^16 = 65536-1 바이트 까지 가리킬 수 있게 2바이트로 설계했습니다.

**사용자 질의 처리(Query Processing)**

사용자가 입력할 수 있는 쿼리는 크게 3가지로 나눌 수 있습니다.

1. **CREATE TABLE**

CREATE TABLE은 다음과 같이 합니다.

CREATE TABLE tableName(student name CHAR(50) grade CHAR(3) sentence VARCHAR(100)

사용자의 테이블 생성 질의에서 tableName을 추출한 뒤 뒤의 컬럼들을 순차적으로 추출합니다.

그 후 테이블명으로 meta.meta파일에 해당하는 테이블이 어디에 기록될 지 추가해줍니다.

|  |
| --- |
| **Metafile [2022-05-06 00:00:01] 생성일자 기록**  student.meta ./student/ (메타 파일명, 폴더 경로)  player.meta ./player/  university.meta ./university/  3 meta files (메타 파일 개수) |

그리고 student.meta파일을 작성합니다.

|  |
| --- |
| student.meta [2022-05-06 00:00:01] (생성일자 기록)  id CHAR(10)  name VARCHAR(20)  grade CHAR(3)  **0001.slot**  1 SLOTS (총 슬롯 페이지 수)  0 ROWS (총 레코드 수) |

최초 테이블을 생성할 시에는 1번 슬롯을 기본적으로 생성해줍니다.

1. **INSERT INTO**

INSERT into student values("minjoon","A+","hello world this is minjoon","SW")

Insert 쿼리를 입력 받았을 시, 테이블명(student)를 우선 추출하고, 메타 파일을 이용해 해당 테이블의 정보를 가져옵니다. 레코드 삭제가 없다고 가정하면 가장 마지막 슬롯을 가져오고, 해당 슬롯의 헤더를 읽어서 빈 공간이 있으면 삽입하고 없으면 새로운 슬롯을 생성합니다. 그리고 슬롯의 헤더에 레코드의 시작 포인터를 2바이트로 가리키게 합니다. 그 후 메타 파일을 업데이트합니다.

레코드를 생성할 때 만약에 쿼리에 values()안에 null이 포함되어 있다면 몇 번째 컬럼인지 체크해서 그 컬럼에 해당하는 null bitmap의 비트를 1로 만들어줍니다.(null 체크)

1. **SELECT**

SELECT \* FROM student WHERE name = "minjoon" AND grade = "A+"

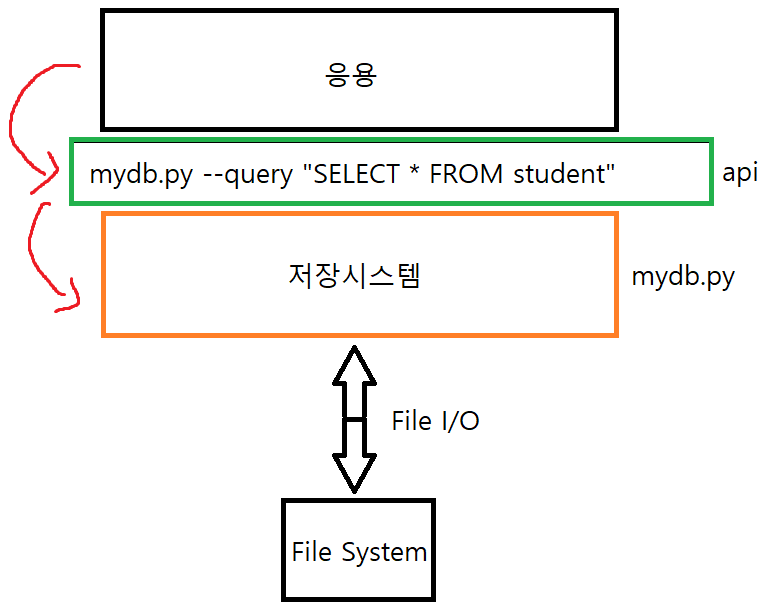
위와 같은 질의를 입력 받았을 경우 meta.meta파일을 읽어 student.meta 파일이 어느 경로에 있는 지 읽어옵니다.

그 후 student.meta파일을 읽어서 student테이블에 관련된 레코드가 있는 모든 슬롯 페이지의 리스트를 획득합니다.

우리의 DBMS에는 인덱스가 구현되어 있지 않기 때문에, 모든 슬롯을 반드시 확인해야 됩니다.

그 후 슬롯 단위로 하나씩 버퍼에 읽어들인 후 WHERE 절에 해당하는 조건의 레코드 인지 모든 레코드를 체크해서, 해당하는 레코드를 버퍼에 남겨놓습니다.

하나의 슬롯을 모두 읽었으면 다음 슬롯으로 이동해서 같은 작업을 반복해서, WHERE 절에 해당하는 레코드를 리턴합니다.



[사진] mydb.py 생성 후 질의 실행 후 파일 I/O를 나타낸 그림

전체적인 그림으로 보면 위와 같은 시스템이 구축이 됩니다.