프로젝트 설계서

프로젝트명 : Turbo C로 구현된 데이터베이스 시스템을 JAVA로 마이그레이션

작성자 : 최진서

1. 프로젝트 개요

1.1 프로젝트 명

C 로 구현된 데이터베이스 시스템을 JAVA 로 마이그레이션

1.2 개발 동기

기존에 C 로 구현되어 있는 데이터베이스 시스템은 Turbo C 환경과 지금은 거의 사용되지 않는 비표준 라이브러리를 필수적으로 요구한다. 이는 개발당시, 시대적 흐름에 따른 당연한 결과라고 할 수 있을 것이다. 과거에 작성된 이 데이터베이스 프로그램은 데이터 테이블과 데이터 접근 구조가 잘 설계되어 있고 특정 필드에 입력된 값의 크기를 기준으로 레코드를 정렬하는 기능이 구현되어 있기 때문에 현재 개발중인 온라인게임 실시간 사용자 ranking 앱의 데이터베이스 시스템에 적합하다고 생각된다. 따라서 개발중인 앱의 동작환경인 Java OpenJDK 13 에서 동작하기 위해 마이그레이션을 진행하고자 한다.

1.3 개발 목적

현재 개발중인 앱과 동일한 자바 표준환경에서 동작하는 데이터베이스 시스템을 처음부터 직접 설계하는 것 보다 기존에 잘 설계된 시스템을 마이그레이션 하는 것이 프로젝트를 진행하는데 시간, 인력 같은 자원관리 측면에서 더 효율적이라고 생각된다. 따라서 비표준환경의 C 로 작성된 코드를 분석하여 시스템의 전체구조를 파악 후, 필요한 부분을 차용하여 Java 환경에 맞게 적절하게 변환한다. 그 과정에서 객체지향 프로그래밍에 적합한 클래스설계와 지원하지 않는 기능에 대해서는 기능을 파악하여 구현하는 것이 요구된다. 마이그레이션을 진행하면서 Java 환경에서 더 효율적으로 동작할 수 있는 방법을 찾고 기존의 시스템을 개선하고자 한다.

1.4 향후 발전 방향

2020 년 현재 가장 많이 사용되는 프로그래밍 언어는 Java 이다. Java 로 작성된 프로그램은 Jvm 위에서 동작하기 때문에 Jvm 만 설치되어 있으면 하드웨어, 운영체제의 종류를 불문하고 실행이 가능한 높은 이식성을 가지고 있으며 많은 사람들이 사용하기

때문에 다양한 기능을 지원하는 라이브러리가 있다. 또한 안드로이드 기반으로 동작하는 모바일 앱은 Java 로 개발되고 있다. Java 로 마이그레이션된 데이터베이스 시스템은 향후 java 를 기반으로 하는 모바일 앱, 또는 서비스를 개발할 때 데이터관리의 필요성을 느낄 때 유용하게 사용할 수 있을 것으로 생각된다.

1.5 개발환경 및 도구

OS: Microsoft Windows10 Enterprise x64,

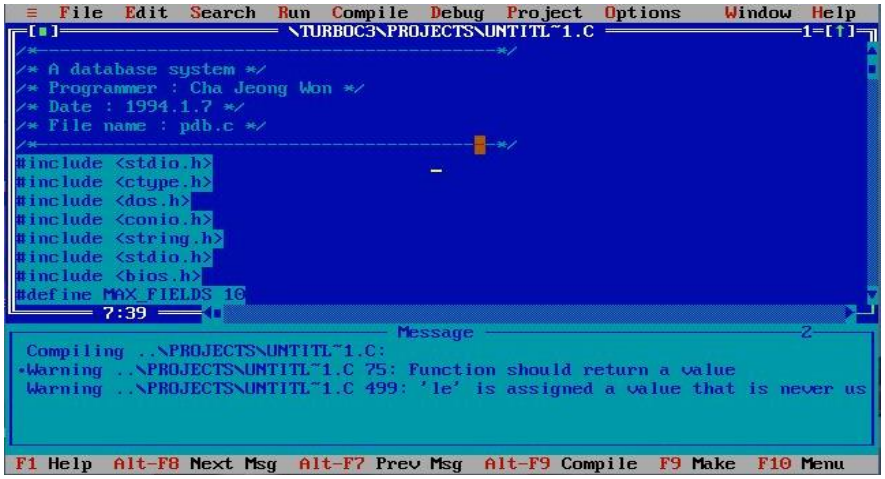
Language: Java(OpenJDK 13), C

개발도구: Microsoft Visual Studio Code, TurboC++ 3.2

2. 배경 지식

2.1 Turbo C

Turbo C 는 1987 년 미국의 Boland 社에서 개발한 C 통합개발환경(IDE)이다.



[그림 2.1 TurboC 실행화면]

당시 Microsoft 의 MS-DOS 환경에서 동작했으며 작은 용량, 낮은 가격, 그리고 빠른 complie 속도를 가지고 있었기 때문에 터보 C 라고 불렸다. 1980 년대 초반 Boland 社는 이미 Turbo Pascal 로 유명세를 타고 있었는데 Turbo C 를 출시하면서 더 큰 성공을 거두게 되었다. 특히 합리적인 가격으로 전문 프로그래머부터 학생까지, 많은 사랑을 받았다. 이후 1990 년 C++를 지원하는 Turbo C++가 출시되고, 1992 년 C++표준이 등장했다. 하드웨어와 운영체제가 지속적으로 발전하고 여러가지 아키텍처가 등장하면서, 기술표준을 위해 C 라이브러리, C++ 표준이 정해지고 지속적으로 업데이트 되는 반면, Turbo C 환경에서 동작하는 라이브러리는 활발하게 업데이트가 진행되지 않아 표준에서 벗어나게 되고, GCC 같은 오픈소스 컴파일러가 등장하면서 자연스럽게 도태되었다. 현재는 일부 학교에서 교육목적으로 사용하는 것을 제외하고 산업전반에서는 C, C++표준을 따른다.

2.2 비표준 Turbo C 라이브러리

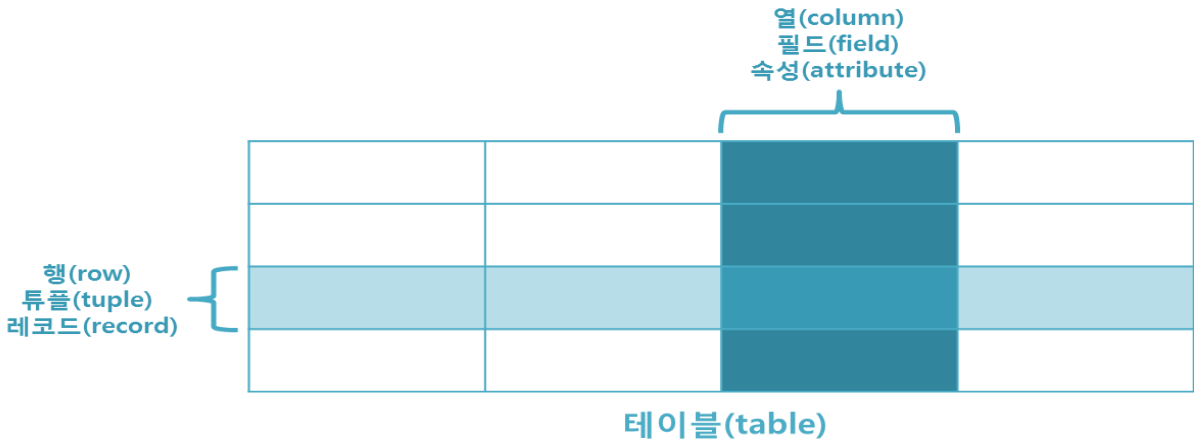
C로 구현된 데이터베이스 프로그램은 Turbo C 환경에서만 동작하는데 이는 프로그램이 동작하는데 필요한 라이브러리가 컴파일러가 가지고 있는 C 표준 라이브러리에 포함되지 않기 때문이다, 프로그램에서 요구하는 비표준 라이브러리에 대한 명세는 다음과 같다.

<bios.h>	BIOS 의 인터럽트를 지원한다. BIOS 는 시스템 리소스로 현재운영체제에서는 시스템 안전을 위해 엄격하게 관리되고 있기 때문에 커널 위에 동작하는 프로그램에서 직접 접근하는 것은 시스템 전체에 risk 를 동반하기 때문에 바람직하지 않다. bios intterpt0x16 를 사용하여 버퍼로부터 키보드입력을 읽어 리턴하는 boiskey(), bios intterpt0x17 을 사용하여 입력받은 파라미터를 출력하는 biosprint()등이 있다.
<conio.h>	콘솔환경에서 입출력을 지원한다. 콘솔에 문자를 입력하는 putch(), 콘솔화면을 초기화하는 clrscr() 등이 있다.
<dos.h>	여러가지 인터럽트(시간,사운드,날짜 등)를 핸들링을 지원한다. 현재시간을 리턴하는 gettime(), 프로그램 실행을 지연시키는 sleep()등이 있다.

최근에 주로 사용되는 GCC, PCC 같은 컴파일러들은 위와 같은 라이브러리를 포함하지 않는다.

2.3 데이터베이스 시스템

데이터베이스란 다수가 공유하여 사용할 목적으로 체계화해 통합, 관리하는 데이터의 집합이다. 중복된 데이터를 없애고 자료를 구조화 하여 효율적인 처리를 할 수 있도록 관리, 운영 되는데 데이터베이스를 관리하는 소프트웨어를 DBMS 라고 한다. DBMS 는 데이터베이스와 사용자 응용프로그램 사이의 중개자로써 모든 응용프로그램들의 데이터베이스 접근을 대행하여 데이터베이스를 관리해주는 소프트웨어 시스템이다. 데이터베이스 시스템은 데이터베이스, 응용프로그램, DBMS 를 통합한 시스템이다. 데이터베이스의 구성요소는 아래와 같다.



[그림 2.3] 데이터베이스 구성요소

그림을 보면 데이터베이스는 테이블들로 이루어져 있으며 이러한 테이블은 key 와 value 의 관계로 나타내어지는데 이러한 구조를 관계형 데이터베이스라고 한다. 테이블은 행과 열로 구성되어 있으며, 거기에 대응되는 값을 가진다. 테이블에서 행(레코드)은 관계된 데이터의 묶음을 의미하고 열(필드)은 유일한 이름을 가지고 데이터의 가장 작은 논리적 단위이다. 관계형 데이터베이스는 테이블들 간의 관계를 맺고 모여 있는 집합체로 말할 수 있다.

3. 유사 서비스

3.1 ORACLE SQL RDBMS

[그림 3.1] ORACLE 社 의로고와 MySQL

현재 가장 많이 사용되고 있는 관계형 데이터베이스 시스템은 ORACLE 社의 RDBMS 와 MySQL 이다. 같은 회사의 제품이지만 MySQL 은 오픈소스라는 특징을 가지고 있다. 위 두 제품은 데이터를 관리하기 위해 설계된 특수목적의 프로그래밍 언어인 SQL 을 사용하여 데이터베이스를 관리하는데 예시는 다음과 같다.

```
--연습문제4. 학생테이블에서 2학년 이상인 학생중에 키가 170 이상인 학생을 모두 출력하라.
SELECT  studno, name, grade, deptno, height, weight
FROM    student
WHERE   grade >='2'
AND     height >='170';
```

질의 결과 ×

SQL | 인출된 모든 행: 6(0.019초)

	STUDNO	NAME	GRADE	DEPTNO	HEIGHT	WEIGHT
1	10101	전인하	4	101	176	72
2	10103	김영균	3	101	170	88
3	10202	오유석	4	102	177	92
4	10105	임유진	2	101	171	54
5	10204	윤진욱	3	102	171	70
6	10107	이광훈	4	101	175	92

[그림 3.2] SQL 작성 예시

위 사진에서 작성된 코드가 SQL 이다. 특정 데이터(레코드)를 조회하기 위해 문법에 맞게 필드에 제약조건과 조회할 테이블을 지정하면 아래에 결과값을 테이블 형태로 출력해

준다. C 로 구현된 데이터베이스 시스템은 데이터를 관리할 때 SQL 쿼리문으로 제어하지 않고 커서와 특정 커맨드로 데이터를 관리한다는 점과 RDBMS 와는 달리 하나하나의 데이터가 다른 데이터와는 별개로 존재하므로 RDBMS 처럼 데이터 간의 관계에 의한 질의 언어가 필요없이 간단하게 Key 만을 검색하면, 그 키에 연결된 레코드를 가져올 수 있고 허용할 수 있는 데이터의 규모가 다르다는 점에서 차이점이 있다.

3.2 GDBM 과 Tokyo Cabinet

[그림 3.3]GNU 와 Tokyo cabinet 로고

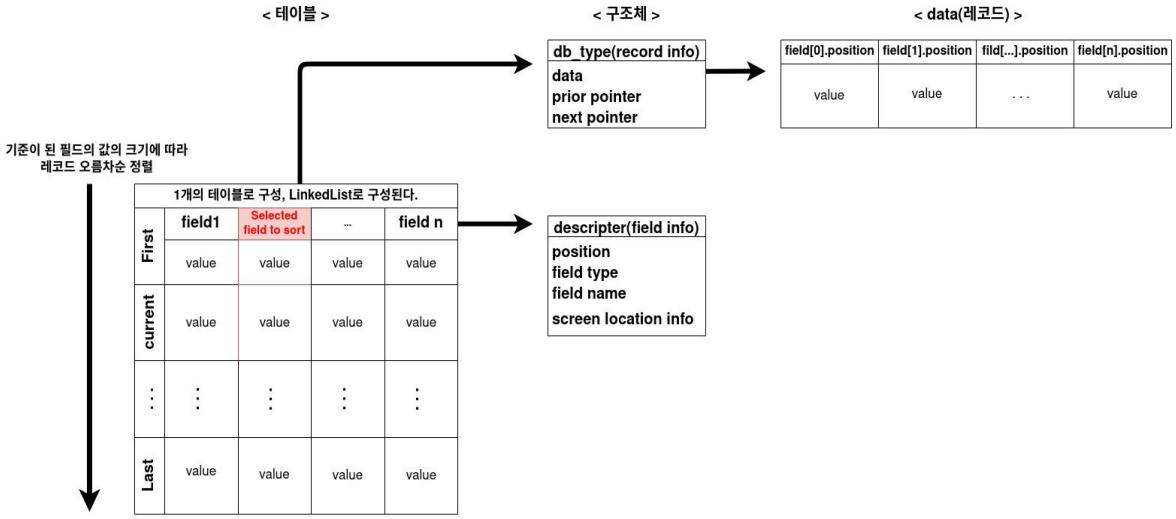
C 로 구현된 데이터베이스 시스템은 다룰 수 있는 데이터의 규모가 작고 SQL 을 사용하지 않는다. 이와 비슷한 기능을 제공하는 GNU 그룹에서 개발한 GDBM 과 Mikio Hirabayashi 가 개발한 Tokyo Cabinet 이 있다. 라이브러리 형태로 제공되며 Tokyo Cabinet 의 경우에는 여러가지 프로그래밍 언어를 지원한다.

```
import kyotocabinet.*;

public class KCDBEX1 {
  public static void main(String[] args) {

    // create the object
    DB db = new DB();

    // open the database
    if (!db.open("casket.kch", DB.OWRITER | DB.OCREATE)){
      System.err.println("open error: " + db.error());
    }

    // store records
    if (!db.set("foo", "hop") ||
        !db.set("bar", "step") ||
        !db.set("baz", "jump")){
      System.err.println("set error: " + db.error());
    }

    // retrieve records
    String value = db.get("foo");
    if (value != null){
      System.out.println(value);
    } else {
      System.err.println("set error: " + db.error());
    }

    // traverse records
    Cursor cur = db.cursor();
    cur.jump();
    String[] rec;
    while ((rec = cur.get_str(true)) != null) {
      System.out.println(rec[0] + ":" + rec[1]);
    }
    cur.disable();

    // close the database
    if(!db.close()){
      System.err.println("close error: " + db.error());
    }

  }
}
```

[그림 3.4]Java 를 지원하는 Tokyo Cabinet 라이브러리로 작성한 데이터베이스 시스템 코드 일부

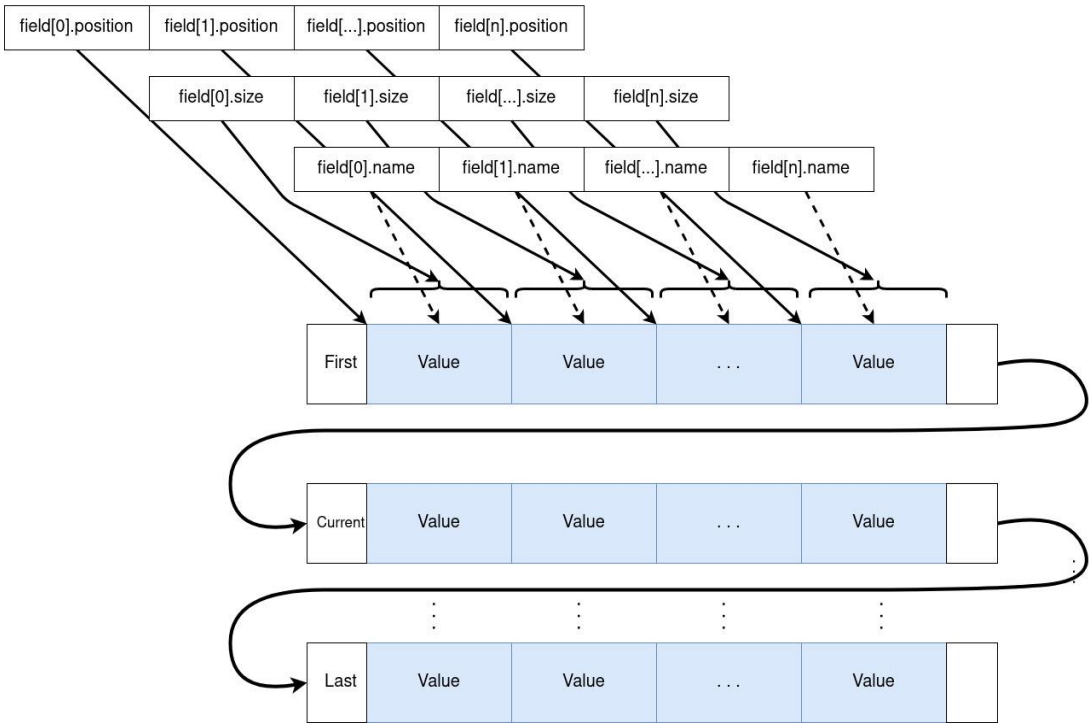
4. 기존 코드 및 구조분석

4.1 데이터베이스 시스템 구조 명세



[그림 4.1 C로 구현된 데이터베이스 시스템 diagram]

4.2 자료구조 명세



[그림 4.2 C로 구현된 데이터베이스 시스템 자료구조 diagram]

4.3 구현된 메소드 명세

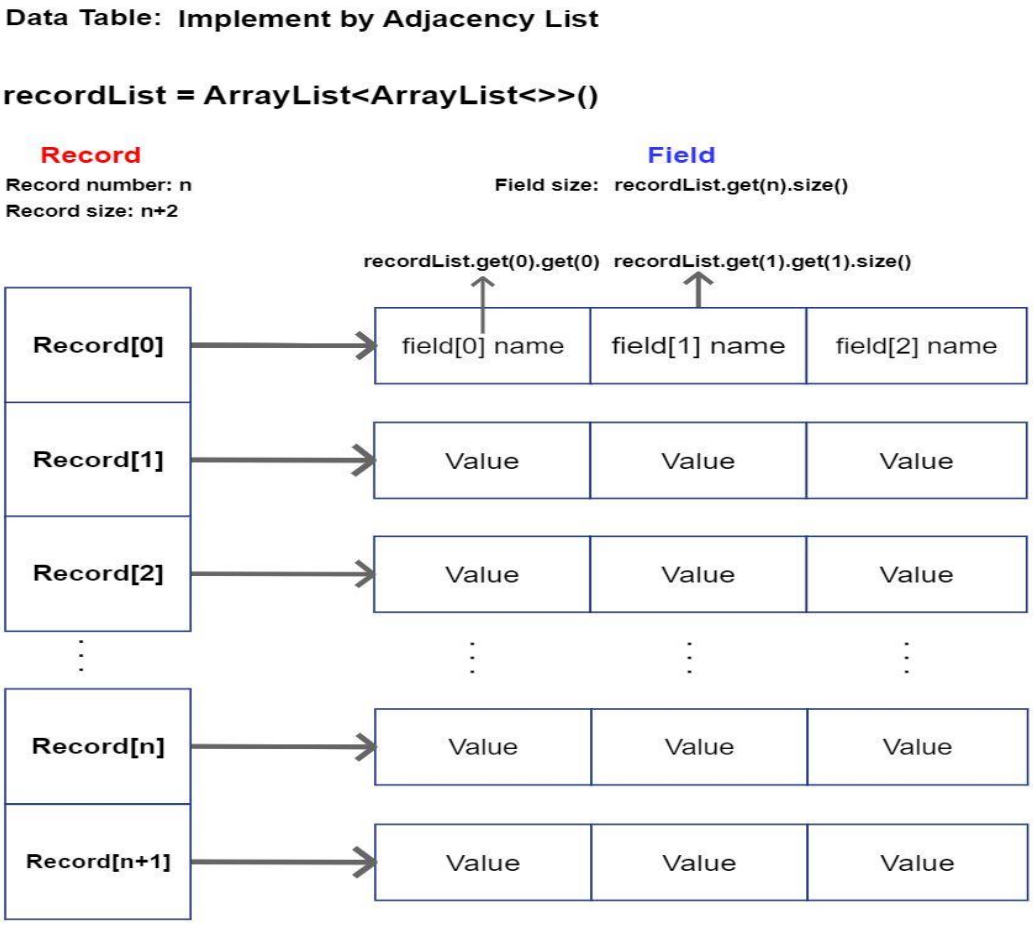
메소드명	설명
`Db(void)`	데이터처리를 위한 기능 선택 메뉴를 콘솔에 출력하고 입력 받은 값에 해당하는 메소드를 호출한다.
`DbGets(char* str, int maxlen)`	키보드로부터 maxlen 만큼의 문자를 입력 받아 *str 으로 저장한다. Enter 키를 입력 받으면 종료되고 str 로 입력받은 문자열을 저장하는데 그 전까지는 Back space 를 입력하여 저장할 문자를 수정할 수 있다.
`ClrEol(int x, int y)`	커서를 입력받은 좌표로 이동시켜 공백으로 한 줄을 초기화한다.
`Define(void)`	사용자로부터 필드의 명세를 입력 받아 필드를 할당하고 레코드 정렬의 기준이 되는 필드를 정한다.
`DisplayRec(char* p)`	파라미터로 입력 받은 레코드의 전체 내용을 콘솔에 출력한다.
`Init(void)`	변수 구조체를 초기화한다.
`DisplayFields(void)`	입력되어 있는 필드의 이름과 크기를 콘솔에 출력한다.
`DlsStore(struct db_type* i, struct db_type** first, struct db_type** last)`	정렬 기준으로 정해진 필드에 입력된 레코드의 크기를 기준으로 파라미터로 입력 받은 레코드 구조체의 data 의 크기를 비교하여 레코드를 오름차순으로 정렬한다.
`Delete(struct db_type** start, struct db_type** last)`	사용자가 필드를 선택하고 삭제될 레코드에 속한 키 값을 입력하면 필드에서 해당하는 키 값을 탐색하여 키 값이 속한 레코드의 삭제 여부를 사용자에게 요청하고 'y' 를 입력 받으면 삭제한 후 레코드를 다시 정렬한다. 만약 키 값이 필드 내에 없으면 종료된다.

`Prompt(char* s)`	파라미터로 받은 문자열을 콘솔에 출력한다.
`Print(void)`	현재 입력 되어있는 레코드를 콘솔에 출력한다. 호출되면 사용자로부터 레코드를 처음부터 출력할 것인지, 특정 레코드부터 출력할 것인지 여부를 요청하고 'y'를 입력 받으면 해당 레코드가 속한 키 값과 출력할 때 레코드 사이의 공백 간격을 입력 받으면 해당 키 값이 속한 레코드부터 콘솔에 한 행씩 출력한다.
`Save(char* fname)`	파라미터로 입력 받은 문자열을 파일 명으로 키 값이 저장되는 .dat 파일, 스키마가 저장되는 .def 파일을 생성하여 Turbo C의 bin 폴더에 파일을 생성한다.
`Load(char* fname)`	파라미터로 입력 받은 문자열에 해당하는 .dat, .def 파일을 불러오는데 호출하기 이전에 입력되어 있는 데이터가 할당된 메모리는 해제하고 불러온 데이터가 저장될 메모리를 새로 할당한다. 파일을 메모리에 적재하면 레코드를 재 정렬한다
`Menu(void)`	콘솔에 사용자를 위한 각 기능에 대한 번호와 설명을 출력한다.
`Enter(void)`	미리 정의되어 있는 필드에 레코드 구조체 메모리를 할당하여 키 값을 입력 받아 레코드 리스트를 만든다. 필드의 개수 만큼 입력 받으면 이대로 입력될 것인지 사용자에게 물어본다 이때 'n'을 입력하면 수정할 레코드를 요청하고 키 값을 수정할 수 있다. define 단에서 정의된 필드의 제약조건에 맞지 않으면 입력되지 않는다. 아무것도 입력하지 않고 Enter 키를 누르면 종료된다.
`SelectField(char* str)`	파라미터로 입력 받은 문자열을 콘솔에 출력하고 사용자로부터 필드를 선택받아 필드의 인덱스 값을 리턴한다.

`Find(char* key, struct db_type* from, int field)`	파라미터로 키 값, 탐색할 구조체, 필드 인덱스를 입력 받으면 필드 인덱스에 해당하는 필드의 키 값과 일치하는 레코드를 찾아서 리턴 한다.
`Search(void)`	사용자로부터 탐색할 필드와 키 값을 입력 받으면 해당 필드에서 키 값과 일치하는 레코드를 콘솔에 출력한다. ‘q’를 누르면 종료되고 찾는 키 값이 없으면 메시지를 출력한다.
`Modify(void)`	사용자로부터 수정할 필드와 키 값을 입력 받으면 키 값과 일치하는 레코드를 찾으면 콘솔에 출력하고 없으면 종료된다. 일치하는 레코드를 찾으면 해당 레코드를 변경할지 여부를 물어본다. ‘y’를 입력하면 키 값의 위치를 변경할 레코드 선택을 요정하고 선택하면 레코드의 필드에 해당하는 키 값의 데이터가 서로 교환된다.
`Browse(void)`	처음 레코드부터 ‘→’, ‘←’, 방향키를 입력 받아 레코드를 콘솔에 출력한다. ‘q’를 입력하면 종료된다.

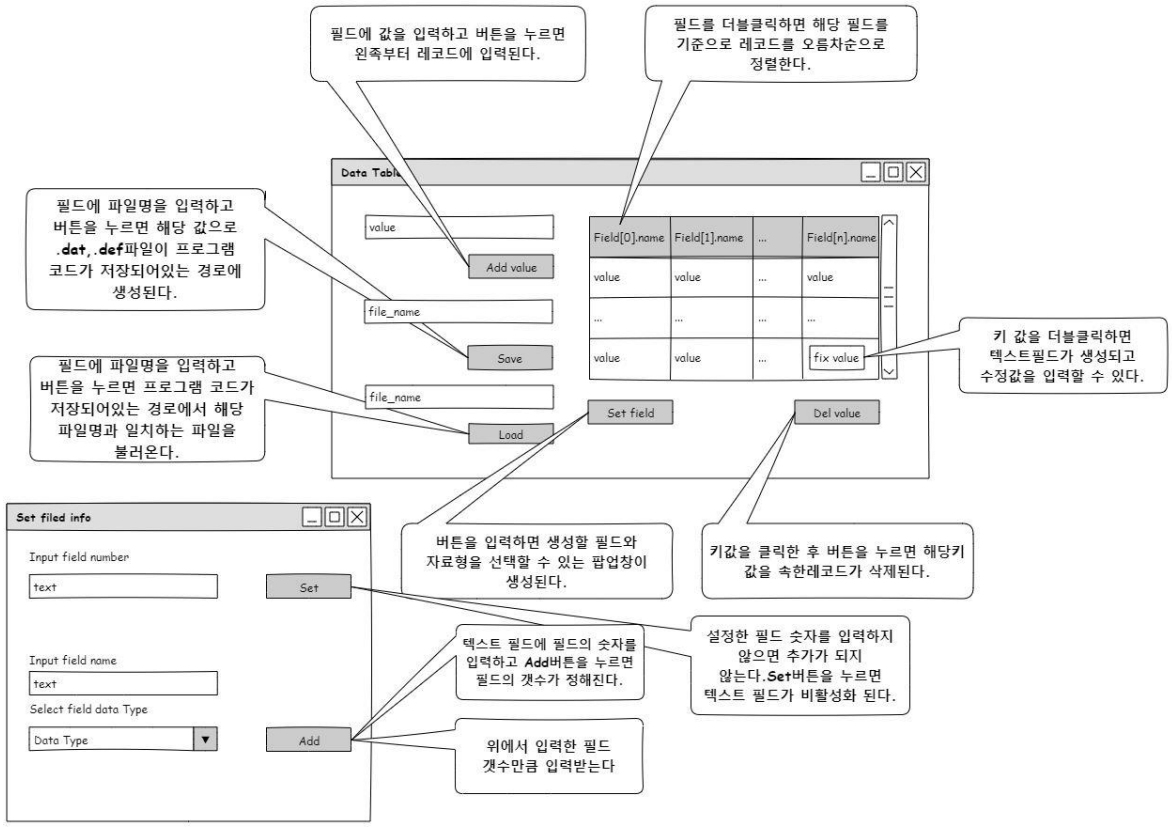
4.3 Java 기반 데이터베이스 시스템 자료구조 설계

기존에 C 로 구현되어 있는 시스템은 테이블 구조를 필드정보를 저장하는 구조체 배열과 레코드를 저장, 관리하는 LinkedList 로 구현하였다. 데이터베이스 시스템은 키 값 접근, 수정, 레코드삭제가 빈번하게 일어나고, 레코드는 필드 값을 기준으로 정렬될 수 있어야 하기 때문에 데이터에 접근이 쉽고 크기가 유연하게 변경될 수 있는 자료구조가 요구되는데 이에 적합한 자료구조가 ArrayList 로 구현한 인접리스트라고 생각된다. Java 는 여러가지 자료구조를 클래스 형태로 제공하여 간편하게 사용할 수 있는 특징이 있는데, ArrayList 클래스는 배열을 리스트형태로 구현한 자료구조로, 데이터의 정렬, 수정, 삭제, 추가 같은 연산이 메소드 형태로 제공된다. 인접행렬을 구현한 데이터테이블 명세는 아래와 같다.



[그림 4.3 인접리스트로 구현한 데이터테이블 명세]

4.4 시스템 GUI mock-up

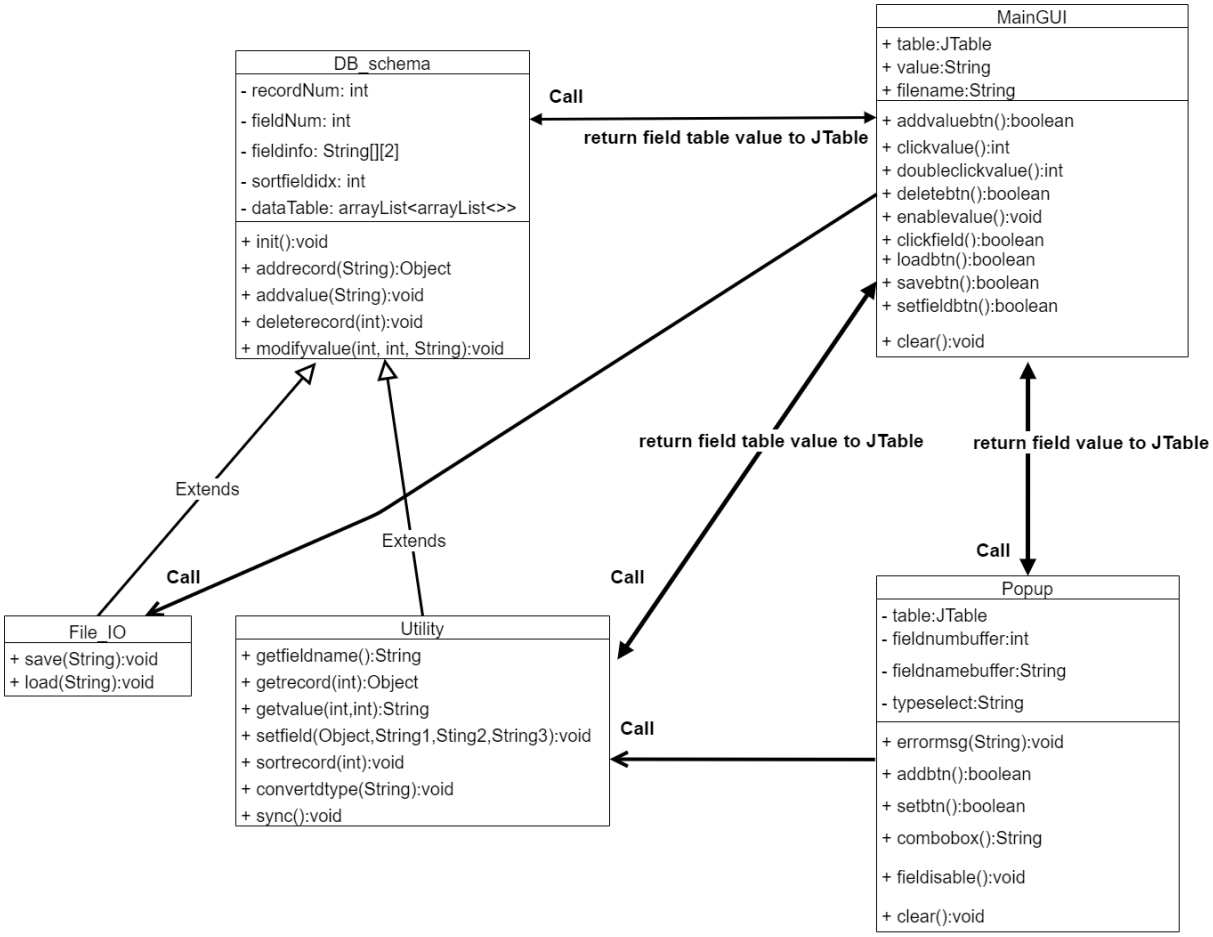


[그림 4.4 데이터베이스 시스템 GUI mock-up]

C 로 구현된 데이터베이스 시스템은 콘솔에서 커맨드와 커서를 사용하는 CLI 를 제공하는데 Java 에서는 커서관련 메소드(예: goto(), clrscr())를 제공하지 않는다. 하지만 GUI 가 등장하며 시대의 흐름에 따라 자연스럽게 CLI 에서 GUI 로 대체된 것처럼 구현할 데이터베이스 시스템도 사용자에게 GUI 를 제공할 것이다. Java 는 GUI 를 구현할 수 있도록 기본적으로 Swing 과 awt 라는 툴킷을 패키지 형태로 제공한다. 코드를 실행하면 위 그림과 같은 팝업창이 생성되면서 GUI 를 사용하여 데이터테이블을 구성하거나 데이터테이블 수정, 삭제, 추가 등의 연산을 수행할 수 있다.

4.5 클래스 구조

구현할 데이터베이스 시스템은 GUI 환경에서 동작하기 때문에 GUI 에서 발생하는 이벤트에 따라 다양할 메소드들이 호출될 수 있도록 설계하였다. 클래스의 구성은 데이터베이스 테이블 자료구조와 테이블에 대한 정보, 값의 추가, 수정, 레코드삭제, 초기화 등 테이블에 관련된 변수와 메소드를 포함하고 있는 클래스인 **DB_schema**, 프로그램 실행 시 처음 팝업되고 필드의 정보를 입력하여 테이블을 생성하는 GUI 구성정보가 포함되어 있는 **Popup** 클래스, 필드정보 입력이 끝나면 팝업되는 메인 GUI 환경에 대한 컴포넌트 구성정보와 이벤트 처리기를 포함하고 있는 **MainGUI** 클래스, 테이블의 필드, 레코드, 키 값을 리턴하거나, 레코드 정렬, 필드 재설정 등의 메소드를 포함하고 있는 **Utility** 클래스, 마지막으로 파일 입출력 메소드를 포함하고있는 **File_IO** 클래스로 전체 시스템이 구성되어 있다. 클래스 구조도는 아래와 같다.



[그림 4.5 데이터베이스 시스템 클래스 UML]

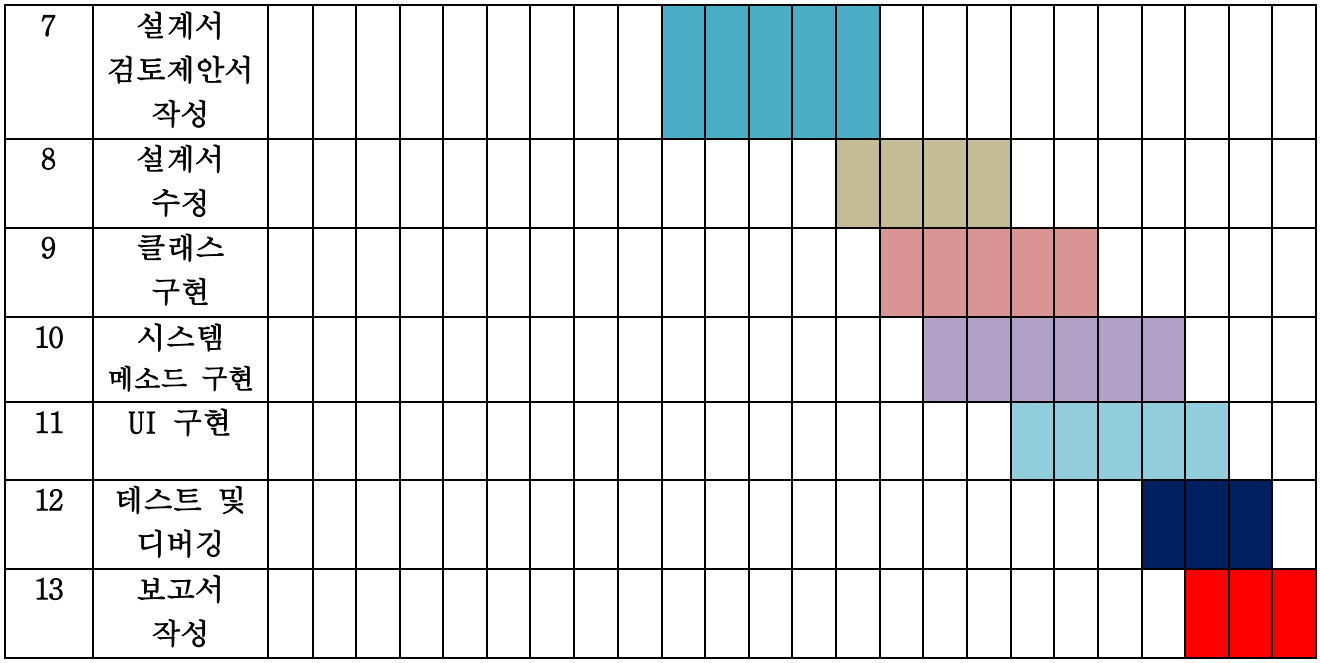
4.6 주요 메소드 추상자료형

클래스	메소드명	설명
DB_schema	Public void Init()	클래스 내 모든 필드를 초기화 한다.
	Public ArrayList addrecord()	데이터 테이블에서 새로운 값이 입력될 레코드의 메모리를 할당한다. 인접리스트에 ArrayList 객체를 생성하여 할당한다.
	Public void addvalue(String)	데이터 테이블 레코드에 파라미터로 입력 받은 값을 추가한다. 바로 이전 인덱스+1 의 값과 필드의 fieldinfo[바로이전 인덱스+1][1]의 값을 참조하여 자료형을 확인하고 자료형에 맞게 변환하여 저장한다.
	Public void deleterecord(int)	레코드의 인덱스를 파라미터로 입력 받아 해당 레코드를 삭제한다. 삭제되면 레코드 정렬을 실행한다.
	Public void modifyvalue(int,int,String)	레코드와 필드의 인덱스 값, 수정할 데이터 값을 파라미터로 입력 받고 해당 값으로 변경한다 이때 fieldinfo[][1]를 참조하여 자료형을 확인하고 자료형에 맞게 변환하여 저장한다.

Utility	Public ArrayList getrecord(int)	레코드의 인덱스값을 파라미터로 입력받아 필드를 객체형태로 반환한다.
	Public void setfield(String,1, String2, String3)	Popup GUI 에서 입력받은 값(필드갯수, 필드이름, 필드 자료형)을 파라미터로 입력받아 클래스 필드에 저장한다.
	Public void sortrecord(int)	입력받은 파라미터는 데이터 테이블에서 필드의 인덱스 값으로 해당 인덱스에 입력된 값을 기준으로 레코드를 오름차순으로 정렬한다.
	Public void convertdtype(String,String)	자료형과 초기 입력데이터를 파라미터로 받아서 fieldinfo[][1]의 자료형 과 일치하는 문자열을 찾아 데이터를 해당 자료형으로 변환한다.
	Public void sync()	JTable 의 객체에 입력 되어있는 값과 DB_schema 클래스의 자료구조에 저장된 데이터를 동기화 한다. 이때 우선순위는 DB_schema 이다.
File_IO	Public void save(String)	입력받은 파라미터 값으로 파일을 생성하여 프로그램 코드가 있는 폴더에 저장한다. 파일은 2 개가 생성되며 하나는 데이터스키마, 다른 하나는 레코드 값이 저장 되어있다. 확장자는.txt 이다.
	Public void load(String)	프로그램코드가 저장된 폴더로부터 문자열과 일치하는 파일을 읽어들여 데이터 테이블에 로드시킨다.

5.개발 일정

순서	기간	9.9. ~9.12.				9.14. ~9.18.					9.21. ~9.25					9.28. ~10.2					10.5. ~10.7.				
	작업내용	1	2	3	4	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	C 코드분석 및 자료조사																								
2	전체 구성요소 설계																								
3	GUI 설계																								
4	추상 자료형 설계																								
5	설계서 작성 및 보고																								
6	설계서 동료평가																								



6. 체크포인트 리스트(Check Point List)

6.1 DB_schema

클래스 내 필드는 전부 private 로 선언하여 메소드를 통해서만 참조할 수 있게 구현한다. Fieldname[][2]배열은 테이블의 필드이름과 자료형을 저장하는 변수이다. 이 값을 **MainGUI** 의 JTable 객체의 생성자로 전달하면 테이블이 되도록 구현한다.

6.2 Popup

제일 처음 프로그램을 실행하면 가장 먼저 팝업 되는 GUI 이다. 사용자로부터 필드 정보를 입력 받는데 처음 텍스트 필드에 필드의 개수를 정수로 입력 받고 Set 버튼을 누르면 텍스트 필드가 비활성화 되도록 구현한다. 그 다음부터 각 필드의 이름과 자료형을 정할 때 한번 Add 버튼을 누를 때 마다 한몫에 값들이 리턴 되어 **DB_schema** 의 필드 정보에 입력이 되도록 하고 Add 버튼을 누르면 텍스트 필드가 초기화 되도록 구현한다.

6.3 Utility

테이블에 관한 각종 정보를 가공하여 리턴 하는 메소드들로 구성된 클래스로 MainGUI 의 이벤트 처리기로부터 리턴 되는 값들을 잘 매칭시켜 구현하는 것이 중요하다. **DB_schema** 를 상속받는 자식클래스로 메소드로부터 리턴 되는 값은 부모클래스의 필드로 입력, 수정된다. 특히 레코드 정렬하는 메소드를 구현할 때 필드 값을 기준으로 정렬하는데 인접리스트의 인덱스 설정이 중요할 것이다.

6.4 MainGUI

Java 에서 지원하는 여러가지 GUI 패키지를 사용하는데 특히 JTable 은 사용자에게 가독성이 좋은 데이터테이블을 제공한다. 객체 생성자로 입력 받는 값으로 테이블을 생성하며 이벤트 처리기로 간편하게 수정, 데이터 조회가 가능하기 때문에 매우 유용하다, 하지만 데이터베이스 시스템 특성상 사용자에 의해 지속적으로 값이 변경되므로 메소드를 통해 생성자에 파라미터로 입력되는 값이 실시간으로 변경될 수 있도록 구현하는 것이 중요하다.

6.5 File_IO

파일은 두개가 생성되는데 하나는 데이터베이스 스키마이고, 다른 하나는 현재 레코드의 값이 저장된다. Utility 클래스의 메소드를 통해 스키마에 입력될 값을 추출하는 메소드를 구현하여 파일에 저장해야 한다. 저장하거나 읽어 들일때는 텍스트 필드에 입력 받는 문자열과 일치하는 파일명을 소스코드가 저장되어 있는 경로에서 탐색한다. 찾는 파일이 없을 때 경고메시지를 출력하도록 구현한다.

8. 자료 및 참고 문헌 출처

[목차 2.1 Turbo C]

https://en.wikipedia.org/wiki/Borland_Turbo_C (Google Wiki 백과)
[그림 2.1] Turbo C 실행화면: 자체 제작

[목차 2.2 비표준 Turbo C 라이브러리]

https://www.programmingsimplified.com/c/dos.h/sleep (Program Simplify)
https://en.wikipedia.org/wiki/Conio.h (Google Wiki 백과)

[목차 2.3 데이터베이스 시스템]

https://www.oracle.com/database/what-is-database.html (Oracle 'what is database')
[그림 2.3] http://www.tcpschool.com/mysql/mysql_intro_relationalDB

[목차 3. 유사 서비스]

[그림 3.1]
https://cyr9210.github.io/2018/11/26/OracleDB/oracle01/
[그림 3.2]: 자체 제작
[그림 3.3]

https://www.gnu.org.ua/software/gdbm/
https://dbmx.net/kyotocabinet/
[그림 3.4]
https://dbmx.net/kyotocabinet/javadoc/
[그림 3.5], [그림 3.6], [그림 3.7]
http://blog.naver.com/many1976?Redirect=Log&logNo=50101854342

[목차 4. 개발내용]

[그림 4.1], [그림 4.2], [그림 4.3], [그림 4.4], [그림 4.5]: 자체 제작