Database System

Spring, 2020

School of Software, CAU

Project

대용량 관계 DB 기반 응용 개발 프로젝트

[도서관 데이터베이스 시스템 구현 보고서]

학과 : 소프트웨어학과

학번 : 20150804

이름 : 정민준

1. Project 개요 :

- 도서관 책 대여 DB 시스템을 구현하였습니다.

- 제공하는 기능은 검색, 삽입, 수정, 삭제로 총 4가지 입니다.

- Table은 student, book, rentbook, department 로 총 4개로 구성되어 있습니다.

- student, book, rentbook 에는 만개 이상의 데이터가 저장되어 있습니다. 각 테이블의 PK, FK 제약사항을 준수하고 특정 조건하에 난수생성을 통하여 데이터를 생성하고 저장하였습니다.

- 트랜잭션 처리는 기능 수행시 이전 savepoint로 선언 후 만약 에러가 발생할 시에 savepoint로 rollback하도록 구현하였습니다.

- 트랜잭션 처리의 중심은 rentbook이 되겠습니다. rentbook 테이블에 기능을 수행할 때 book와 student 테이블을 참조하고 있으며 두 테이블의 변경 상황에 맞춰 잘 작동하도록 구현하였습니다.

2. Project DBMS 및 대화식 SQL 도구 :

- MySQL를 기반으로 하였습니다.

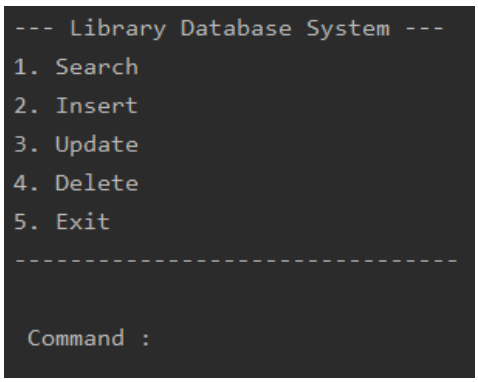
- JDBC 구현은 intellij 코드 에디터를 사용하였고 MySQL 8.0 Command line Client와 연동하였습니다.

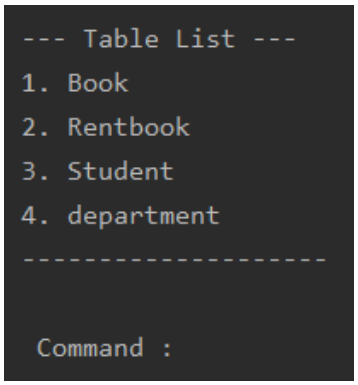
3. 주언어 및 DB API :

- Java / JDBC

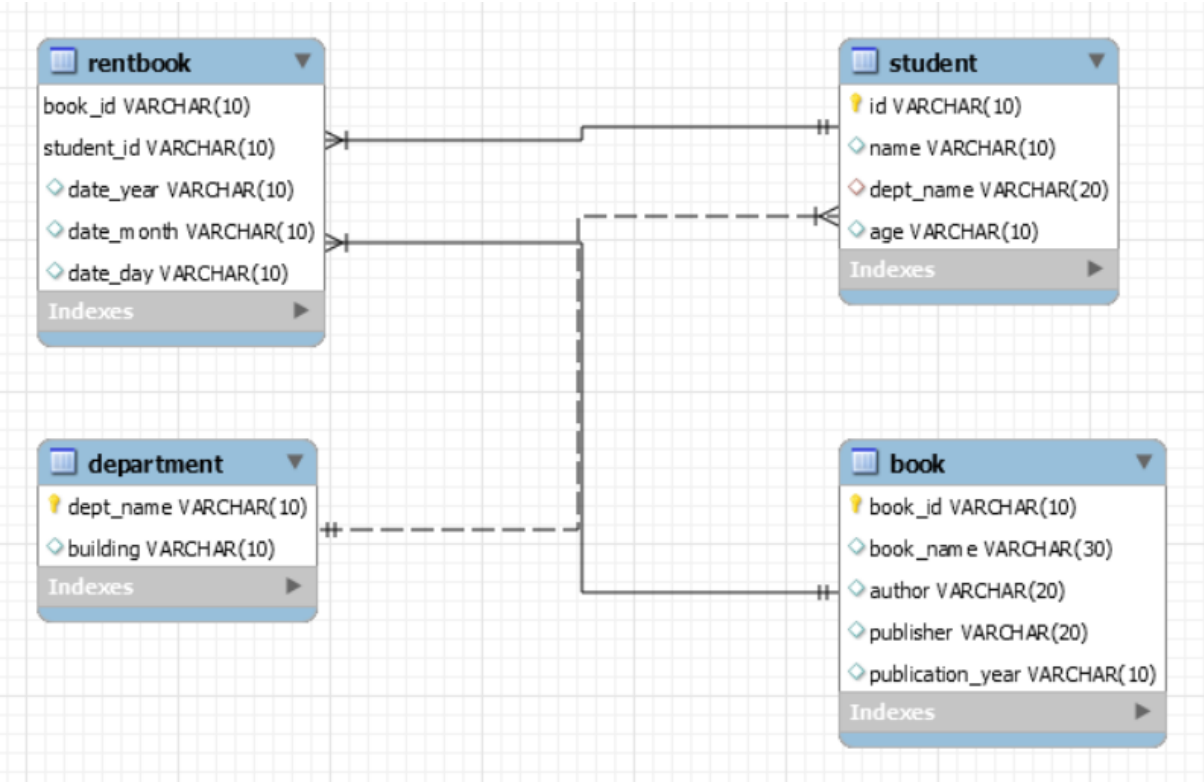
4. UI :

- Console application을 통한 interface를 구현하였습니다.





5. 관계 DB 스키마 :



6. 작동 방식 및 Pagination 구현 소개 :

- 작동방식은 다음과 같습니다.

1) 먼저 수행할 기능을 선택합니다. 올바르지 않은 기능을 선택시 다시 선택하도록 합니다.

2) 선택한 기능을 수행할 테이블을 선택합니다. 올바르지 않은 기능을 선택시 다시 선택하도록 합니다.

3) 선택한 기능이 추가작업을 요구할 경우 그에 해당하는 입력을 추가로 합니다. 여기서 제약사항에 맞지않는 작업이 발생시 에러를 발생시키고 처음 화면으로 돌아갑니다.

4) 기능이 수행되었을 때 결과를 100개 단위로 출력합니다. 만약 페이지를 넘기고 싶다면 원하는 페이지 번호를 입력합니다. 조회를 끝내고 싶다면 "-1"를 입력합니다.

5) 수행할 기능을 선택하는 화면으로 되돌아옵니다. 종료 기능을 선택시 프로그램이 종료됩니다.

- **검색, 삽입, 수정, 삭제, 종료순서로 설명하겠습니다.**

1. Select

- student table의 레코드를 검색하는 기능을 수행하겠습니다. 다음과 같이 1, 3을 입력으로 하여 기능은 select, 대상 table은 student로 합니다.

```
--- Library Database System ---
1. Search
2. Insert
3. Update
4. Delete
5. Exit
--------------------------------

 Command :
1

--- Table List ---
1. Book
2. Rentbook
3. Student
4. department
--------------------

 Command :
3
```

- 다음 수행 결과로 student table에 대한 설명과 함께 레코드를 확인할 수 있습니다.

```
--- Student Table ---
1. student_id (PK)
2. name
3. dept_name
4. age

 -- Student Table Result Set --
 -- Now Page : 0 --
-----------------------
20150804, minjoon, Comp.sci, 25
S0, Tom0, Economics, 22
S1, Harry1, Biology, 22
S10, Adam10, Biology, 24
S100, Tom100, Comp.sci, 27
S1000, Tom1000, Physics, 20
S10000, Max10000, Biology, 23
S10001, Max10001, Comp.sci, 28
S10002, Bob10002, Comp.sci, 21
```

- 100개 단위로 페이지 출력을 해서 끝부분을 따로 찍었습니다. 다른 페이지를 호출하는 과정입니다.

```
S10081, Bob10081, English, 22
S10082, Min10082, Economics, 29
S10083, Max10083, English, 24
S10084, Adam10084, English, 28
S10085, Harry10085, Biology, 28
총 페이지 수 : 130
검색결과를 마무리려면 '-1' 를 입력하세요.
페이지 번호 :100

 -- Student Table Result Set --
 -- Now Page : 100 --
----------------------
S7298, Harry7298, English, 26
S7299, Min7299, Physics, 21
S73, Harry73, Physics, 26
S730, Bob730, Comp.sci, 24
S7300, Tom7300, Physics, 26
S7301, Tom7301, English, 23
S7302, Min7302, Economics, 26
S7303, Max7303, Economics, 27
```

- 다음과 같이 100번 페이지를 호출한 결과를 확인할 수 있습니다.

2. Insert

- student table에 제 정보를 삽입하겠습니다. 이어서 제약사항을 어겼을 경우도 처리하였습니다. 순서대로 설명하겠습니다.

```
--- Student Table ---
1. student_id (PK)
2. name
3. dept_name
4. age
student_id :20150804
student_name :minjoon
dept_name :Comp.Sci
age :25
```

- 다음과 같은 삽입 연산을 수행시에 이미 20150804 id를 가진 학생이 존재하기에 "PK 입력값을 다시 확인해주세요." 라는 메시지와 함께 기능 선택화면으로 돌아가게 됩니다.

```
mysql> select * from student where id = "20150804";
+----------+---------+-----------+------+
| id       | name    | dept_name | age  |
+----------+---------+-----------+------+
| 20150804 | minjoon | Comp.sci  | 25   |
+----------+---------+-----------+------+
1 row in set (0.00 sec)
```

3. Update

- Rentbook 테이블에 update 문을 실행하겠습니다. 실행과정과 실행 전, 후의 사진입니다.

```
mysql> select * from rentbook where book_id = "CAU10270" and student_id = "S11515";
+---------+------------+-----------+------------+----------+
| book_id | student_id | date_year | date_month | date_day |
+---------+------------+-----------+------------+----------+
| CAU10270 | S11515     | 2016      | 5          | 22       |
+---------+------------+-----------+------------+----------+
1 row in set (0.00 sec)

mysql> select * from rentbook where book_id = "CAU10270" and student_id = "S11515";
+---------+------------+-----------+------------+----------+
| book_id | student_id | date_year | date_month | date_day |
+---------+------------+-----------+------------+----------+
| CAU10270 | S11515     | 1996      | 5          | 7        |
+---------+------------+-----------+------------+----------+
1 row in set (0.00 sec)
```

```
--- Rentbook Table ---
1. book_id (PK)
2. student_id (PK)
3. date year
4. date month
5. date day
year :1996
month :5
day :7
book_id :CAU10270
student_id :S11515
입력사항으로 갱신하였습니다.
```

- 오른쪽 사진은 update문 실행에 입력값 입니다. 왼쪽사진에서 갱신 전 select문 결과의 date_year, daye_month, date_day가 실행 후 갱신되었음을 확인할 수 있습니다.

- 이어서 student 테이블에도 update문을 실행하겠습니다.

```
mysql> select * from student where id = "S8876";
+-------+-----------+-----------+------+
| id    | name      | dept_name | age  |
+-------+-----------+-----------+------+
| S8876 | Harry8876 | Comp.sci  | 29   |
+-------+-----------+-----------+------+
1 row in set (0.00 sec)

mysql> select * from student where id = "S8876";
+-------+---------+-----------+------+
| id    | name    | dept_name | age  |
+-------+---------+-----------+------+
| S8876 | minjoon | English   | 25   |
+-------+---------+-----------+------+
1 row in set (0.00 sec)
```

```
--- Student Table ---
1. student_id (PK)
2. name
3. dept_name
4. age
student name :minjoon
dept_name :English
age :25
student_id :S8876
입력사항으로 갱신하였습니다.
```

- 오른쪽 사진의 입력값으로 잘 갱신되었음을 확인할 수 있습니다.

4. Delete

- rentbook 테이블에 delete 문을 실행하겠습니다. Book_id 가 "CAU10304", student_id 가 "S2249" 인 학생의 대여기록을 삭제하겠습니다.

```
mysql> select * from rentbook where book_id = "CAU10304" and student_id = "S2249";
+---------+------------+-----------+------------+----------+
| book_id | student_id | date_year | date_month | date_day |
+---------+------------+-----------+------------+----------+
| CAU10304 | S2249      | 2016      | 8          | 23       |
+---------+------------+-----------+------------+----------+
1 row in set (0.00 sec)

mysql> select * from rentbook where book_id = "CAU10304" and student_id = "S2249";
Empty set (0.00 sec)
```

```
--- Rentbook Table ---
1. book_id (PK)
2. student_id (PK)
3. date year
4. date month
5. date day
book_id :CAU10304
student_id :S2249
입력사항으로 갱신하였습니다.
```

- 오른쪽 사진과 같이 입력값에 해당하는 레코드가 삭제되었음을 왼쪽사진 select 문의 결과로 확인할 수 있습니다.

- 이어서 Department 테이블에 delete 문을 실행한 결과를 확인하겠습니다.

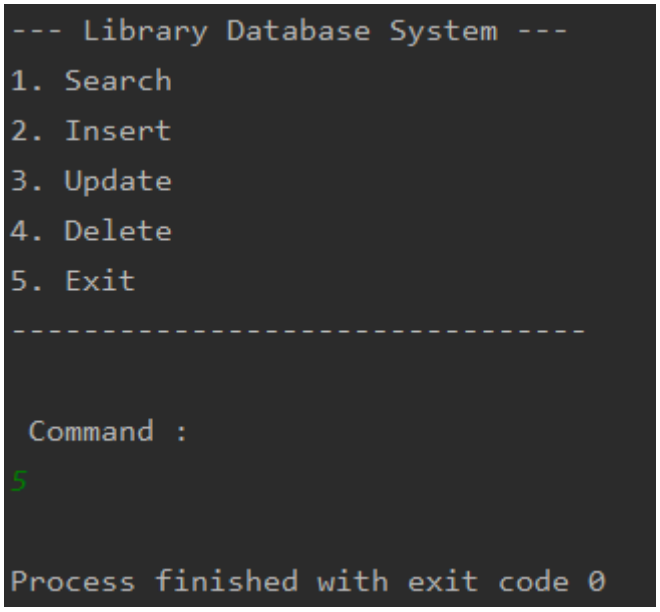
```
mysql> select * from department;
+-----------+----------+
| dept_name | building |
+-----------+----------+
| barista   | 310      |
| Biology   | 102      |
| Comp.sci  | 101      |
| Economics | 104      |
| English   | 103      |
| Physics   | 100      |
+-----------+----------+
6 rows in set (0.00 sec)

mysql> select * from department;
+-----------+----------+
| dept_name | building |
+-----------+----------+
| Biology   | 102      |
| Comp.sci  | 101      |
| Economics | 104      |
| English   | 103      |
| Physics   | 100      |
+-----------+----------+
5 rows in set (0.00 sec)
```

```
--- Department Table ---
1. dept_name (PK)
2. building
dept_name : barista
입력사항으로 갱신하였습니다.
```

- "barista" 학과가 삭제되었습니다.

5. Exit

- 종료를 수행한 사진입니다.



7. 트랜잭션 설계 :

1. 구현

1) Rentbook table select transaction :

rentbook 은 book 와 student 테이블의 pk 를 참조하고 있습니다. rentbook 에서 select 문 수행시 book 와 student 테이블의 레코드 변경에 반응해야 합니다. 따라서 rentbook 테이블에 select 문 수행전 savepoint 를 지정합니다. 그리고 수행 후 에러 발생시 savepoint 로 rollback 할 수 있도록 구현하였습니다.

2) Book table select transaction :

Book 테이블에 select 문 수행중 레코드가 변경되었다면 rollback 후 다시 select 문을 수행할 수 있도록 구현하였습니다.

3) Book table update transaction :

Book 테이블에 update 문 수행중 레코드가 변경혹은 삭제가 된다면 해당 update 문은 일어나지 않은 상태로 되돌아가야 합니다. 그러므로 수행전 savepoint 를 저장하고 에러 발생시 rollback 할 수 있도록 합니다. (b) 조건

4) Student table update transaction :

Student 테이블 update 문 수행중 department 가 변경, 삭제가 발생하거나 해당 학생의 정보가 수정, 삭제되었을 경우 해당 update 문의 처리가 필요합니다. 따라서 해당 테이블의 update 문 수행 전 savepoint 로 rollback 할 수 있게끔 코드를 구현하였습니다.

5) Department table select transaction :

Department 테이블 select 문 트랜잭션을 설계하였습니다. 검색연산 도중 레코드가 변경되었다면 이전 상태로 rollback 후 다시 select 문을 수행할 수 있게끔 구현하였습니다.

6) Rentbook update select transaction :

Rentbook 테이블 update 문 트랜잭션을 설계하였습니다. Rentbook 테이블 레코드를 변경하는 도중 참조하고있는 book, student 레코드의 삭제가 발생하면 해당 Update 문은 없던 작동이 되어야 하고 만약 변경이 되었다면 변경된 레코드 상태에서 update 문을 실행하여야 합니다. 따라서 update 하고자 하는 레코드가 존재하지않을 경우와 레코드가 변경될 경우를 고려하여 트랜잭션을 설계하였습니다.

7) Student table select transaction :

Student 테이블 select 문 트랜잭션을 설계하였습니다. 검색연산 도중 department 테이블의 PK 인 dept_name 이 변경되었거나 삭제, 그리고 Student 테이블 레코드의 변경, 삭제가 일어난다면 변경사항을 반영한 결과를 select 해야할 것입니다. 그러므로 검색에서 에러가 발생하면 rollback 을 통하여 검색전의 지점으로 이동하여 다시 검색을 할 수 있도록 트랜잭션을 구현하였습니다.

2. 결과

1) select * from student;

- student select 문 transaction, 검색 only

```
--- Student Table ---
1. student_id (PK)
2. name
3. dept_name
4. age

 -- Student Table Result Set --
 -- Now Page : 0 --
-----------------------
20150804, minjoon, Comp.sci, 25
S0, Tom0, Economics, 22
S1, Harry1, Biology, 22
S10, Adam10, Biology, 24
S100, Tom100, Comp.sci, 27
S1000, Tom1000, Physics, 20
S10000, Max10000, Biology, 23
S10001, Max10001, Comp.sci, 28
S10002, Bob10002, Comp.sci, 21
S10003, Min10003, English, 23
S10004, Min10004, Comp.sci, 28
S10005, Max10005, English, 20
S10006, Min10006, Biology, 28
S10007, Max10007, Comp.sci, 20
S10008, Adam10008, Economics, 24
S10009, Harry10009, Comp.sci, 27
```

2) select * from book;

- book select 문 transaction, 검색 only

```
 -- Book Table Result Set --
 -- Now Page : 0 --
-----------------------
, 1, 1, 1, 1
CAU0, B0, author51, Fox, 2028
CAU1, B1, author84, CAT, 2016
CAU10, B10, author6, Fox, 2003
CAU100, B100, author7, CAT, 2018
CAU1000, B1000, author81, CAT, 2014
CAU10000, B10000, author20, Dog, 2029
CAU10001, B10001, author47, Dog, 2028
CAU10002, B10002, author88, Dog, 2014
CAU10003, B10003, author39, CAT, 2019
CAU10004, B10004, author52, Dog, 2023
CAU10005, B10005, author15, Wolf, 2020
CAU10006, B10006, author8, Fox, 2021
CAU10007, B10007, author93, CAT, 2010
CAU10008, B10008, author60, Wolf, 2016
CAU10009, B10009, author33, Wolf, 2009
CAU1001, B1001, author58, CAT, 2022
```

3) Update book set publication_year = “2000” where book_id = “CAU4”;

- 변경 트랜잭션

```
입력사항으로 갱신하였습니다.

Process finished with exit code 0
```

4) Update department set building = ‘110’ where dept_name = ‘physics’;

- 변경 트랜잭션

```
입력사항으로 갱신하였습니다.

Process finished with exit code 0
```

6) select * from department

Update set building = 500 where department = “Comp.sci”;

```
 -- Department Table Result Set --
 -- Now Page : 0 --
------------------------
Biology, 102
Comp.sci, 101
Economics, 104
English, 103
Physics, 100
총 페이지 수 : 0
검색결과를 마무리려면 '-1' 를 입력하세요.
페이지 번호 :
```

```
 -- Department Table Result Set --
 -- Now Page : 0 --
-----------------------
Biology, 102
Comp.sci, 500
Economics, 104
English, 103
Physics, 100
총 페이지 수 : 0
검색결과를 마무리려면 '-1' 를 입력하세요.
페이지 번호 :
```

- 검색, 변경 혼합 트랜잭션

7) select * from student where id = ?

Update set student_name = ?, dept_name = ?, age = 25 where student_id = ?

- 검색, 변경 혼합 트랜잭션

- id 는 S15, name 은 CAUBOY, dept_name 은 Physics, age 는 25 로 하였습니다.

```
mysql> select * from student where id ="S15";
+-----+-------+-----------+------+
| id  | name  | dept_name | age  |
+-----+-------+-----------+------+
| S15 | Max15 | Economics | 29   |
+-----+-------+-----------+------+
1 row in set (0.00 sec)
```

```
student name :CAUBOY
dept_name :Physics
age :25
student_id :S15
입력사항으로 갱신하였습니다.
```

```
mysql> select * from student where id ="S15";
+-----+--------+-----------+------+
| id  | name   | dept_name | age  |
+-----+--------+-----------+------+
| S15 | CAUBOY | Physics   | 25   |
+-----+--------+-----------+------+
1 row in set (0.00 sec)
```

5) select * from rentbook where student_id in (select student_id from student where age = ?);

- rentbook select 문 transaction

- select 문 2 개 이상

- 후행 select 문 동적 매개변수의 값으로 할당 및 복수개 select 문 결과 조합.

- age 는 29 로 지정하였습니다.

```
 -- Rentbook Table Result Set --
-- Now Page : 0 --
-----------------------
CAU348, S1002, 2019, 2, 19
CAU4416, S1002, 2019, 2, 1
CAU8725, S10020, 2016, 3, 23
CAU9305, S10024, 2019, 8, 8
CAU11769, S10036, 2015, 7, 21
CAU2149, S10036, 2018, 6, 4
CAU2481, S10036, 2019, 1, 20
CAU12462, S10049, 2015, 2, 3
CAU2100, S10055, 2019, 9, 15
CAU14839, S10068, 2017, 11, 21
CAU5311, S10142, 2015, 2, 20
CAU13181, S10166, 2016, 4, 5
CAU3133, S10166, 2016, 1, 3
CAU7780, S1018, 2015, 6, 14
CAU2204, S10200, 2015, 2, 20
CAU4262, S10200, 2016, 7, 20
CAU6728, S10200, 2019, 10, 15
CAU3820, S10209, 2018, 11, 18
CAU6016, S1021, 2016, 2, 20
CAU8136, S1021, 2015, 9, 7
```

6) select * from department where dept_name in (select dept_name from student where age = ?);

- department select 문 transaction

- select 문 2 개 이상

- 후행 select 문 동적 매개변수의 값으로 할당 및 복수개 select 문 결과 조합.)

- age 는 25 로 지정하였습니다.

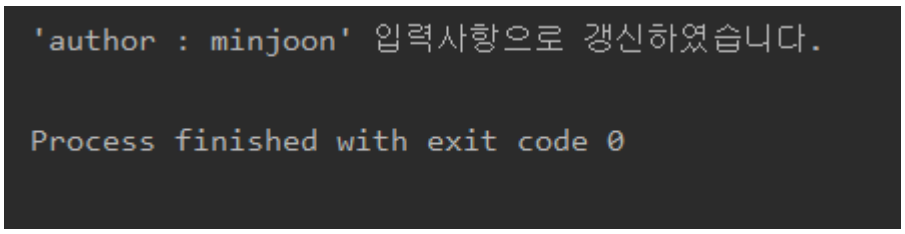
```
 -- Department Table Result Set --
 -- Now Page : 0 --
------------------------
Biology, 102
Comp.sci, 101
Economics, 104
English, 103
Physics, 100
총 페이지 수 : 0
검색결과를 마무리려면 '-1' 를 입력하세요.
페이지 번호 :
```

7) Update book set author = ? where book_id in (select book_id from book where publisher = ?);

- 변경문의 동적 매개변수의 값을 선행 select 문의 결과 셋의 데이터 값 기반으로 할당.

- select 문의 동적 매개변수의 값을 UI 에서 입력 받은 값으로 할당.

- author 은 minjoon, publisher 은 FOX 로 하였습니다.

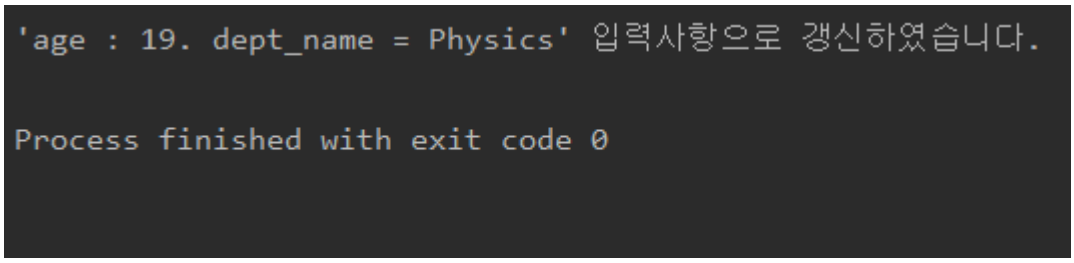


8) Update student set age = ?, dept_name = ? where id in (select student_id from rentbook where book_id = ?);

- 변경문의 동적 매개변수의 값을 선행 select 문의 결과 셋의 데이터 값 기반으로 할당.

- select 문의 동적 매개변수의 값을 UI 에서 입력 받은 값으로 할당.

- age 는 19, dept_name 은 physics, book_id 는 CAU1038 으로 지정했습니다.



9) Update rentbook set date_year = ?, date_month = ?, date_day = ? where student_id in (select id from student where dept_name in (select dept_name from department where building = ?))

and book_id in (select book_id from book where publisher = ?);

- select 문 최소 2 개 이상

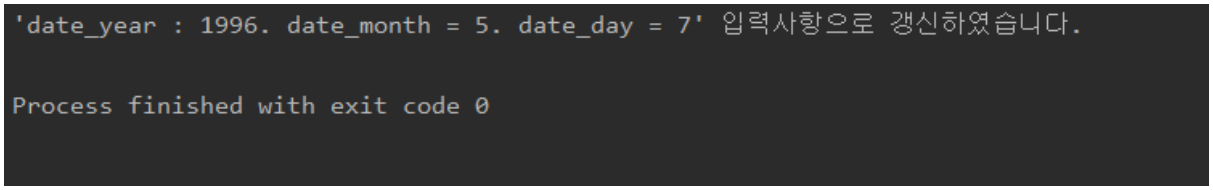
- select 문의 동적 매개변수의 값을 UI 에서 입력 받은 값으로 할당.

- 선행 select 문의 검색 결과 데이터 값을 후행 select 문의 동적 매개변수의 값으로 할당

- 복수개 select 문 결과 조합.

- 변경문의 동적 매개변수 값을 선행 select 문의 결과 셋의 데이터 값에 기반으로 할당.

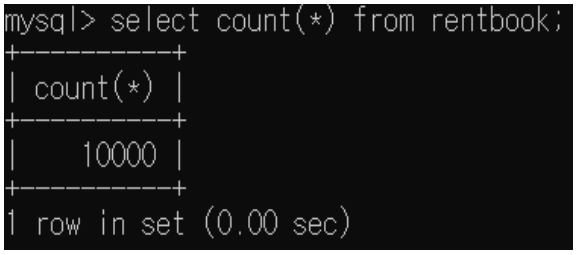
- date_year = 1996, date_month = 5, date_month = 7 로 지정하였습니다. 그외 department select 문의 building 은 "101", book select 문의 publisher 은 "Wolf"로 하였습니다.

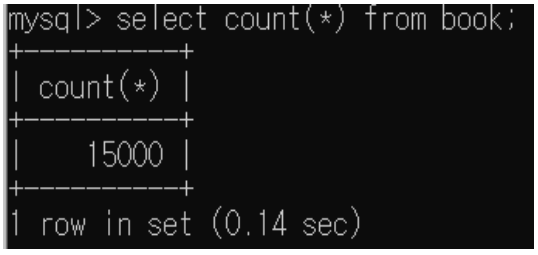


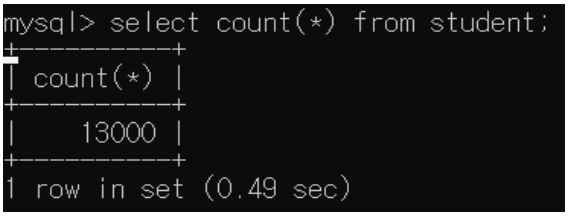
8. 프로젝트 요건 및 테스트 DB에 load한 레코드 수:

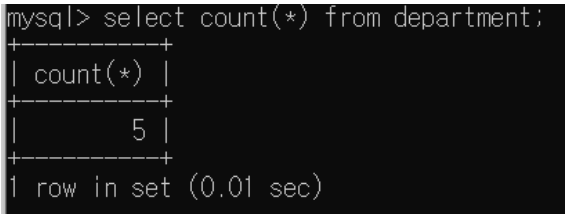
- 먼저 각 테이블에 레코드가 삽입되어야 합니다. 그리고 일정한 제약을 통해 생성되어 적재되어야 하는데 이는 레포트 3 번에 작성하였습니다.

- 레코드 생성은 MakeData.java, 레코드 삽입은 LoadData.java, 프로젝트 실행은 library_DB_System.java 소스코드에 구현되어 있습니다. 코드 실행순서도 다음과 같습니다. 그리고 레코드 수 사진을 첨부하였습니다.









9. 소스코드 설명 및 매뉴얼 :

- 프로젝트 소스코드는 library_DB_System.java 입니다. 앞서 설명한바와 같이 사용자 선택에 의해서 기능, 테이블이 선택되고 수행됩니다. Pagination 기능을 제공하고 console application 에 기반하여 interface 를 구현하였습니다.

- Switch / Case 를 통해 테이블을 구별하였습니다. (0 : book, 1 : rentbook, 2 : student, 3 : department)

- 조건문을 통해 기능을 구별하였습니다. (0 : select, 1 : insert, 2 : update, 3 : delete 4 : exit)

- 이에 해당하지 않는 번호를 입력시 다시 입력을 받도록 while 문을 사용하였습니다.

- 각각 발생할 수 있는 에러에서 코드가 멈추지 않고 에러 발생 메시지와 함께 처음 화면으로 돌아가도록 하였습니다. 코드를 종료할 수 있는 방법은 메인 화면에서 종료기능을 선택하는 것입니다. 그 외의 종료될 수 있는 부분은 예외처리 하였습니다.

```
user_command = InitialPage();
if (user_command > 5 || user_command < 1){
    while (user_command > 5 || user_command < 1 ){
        System.out.println("지원하지 않는 기능입니다. 다시 입력해주세요.");
        user_command = InitialPage();
    }
}

if (user_command == 5) return;

user_table = SelectTablePage();
if (user_table > 4 || user_table < 1){
    while (user_table > 4 || user_table < 1 ) {
        System.out.println("지원하지 않는 기능입니다. 다시 입력해주세요.");
        user_table = SelectTablePage();
    }
}

Main_System(user_command, user_table);
```

<Command 를 올바르게 입력받기 위한 코드구문>

```
// rentbook table
else if (table_idx == 1){
    rentbook_table();

    stmt = (Statement) conn.createStatement();
    RentbookUpdatepoint = conn.setSavepoint();

    switch (command_idx){
        // select
        case 0:
            try{
                RentbookSelectpoint = conn.setSavepoint();
                rs = stmt.executeQuery( sql: "select * from rentbook");
                ArrayList<String> book_id = new ArrayList<>();
                ArrayList<String> student_id = new ArrayList<>();
                ArrayList<String> date_year = new ArrayList<>();
                ArrayList<String> date_month = new ArrayList<>();
                ArrayList<String> date_day = new ArrayList<>();

                while (rs.next()){
                    book_id.add(rs.getString( columnLabel: "book_id"));
                    student_id.add(rs.getString( columnLabel: "student_id"));
                    date_year.add(rs.getString( columnLabel: "date_year"));
                    date_month.add(rs.getString( columnLabel: "date_month"));
                    date_day.add(rs.getString( columnLabel: "date_day"));
```

<Rentbook 테이블의 select 문>

```
    int result = pstmt.executeUpdate();

    if (result == 1) System.out.println("성공적으로 삽입 되었습니다.");
    else System.out.println("입력이 실패했습니다.");
}

// PK 관련 제약사항 위배시
catch (SQLIntegrityConstraintViolationException e){
    System.out.println("PK 입력값을 다시 확인해주세요");
}
```

<book table 의 insert 문 예외처리>