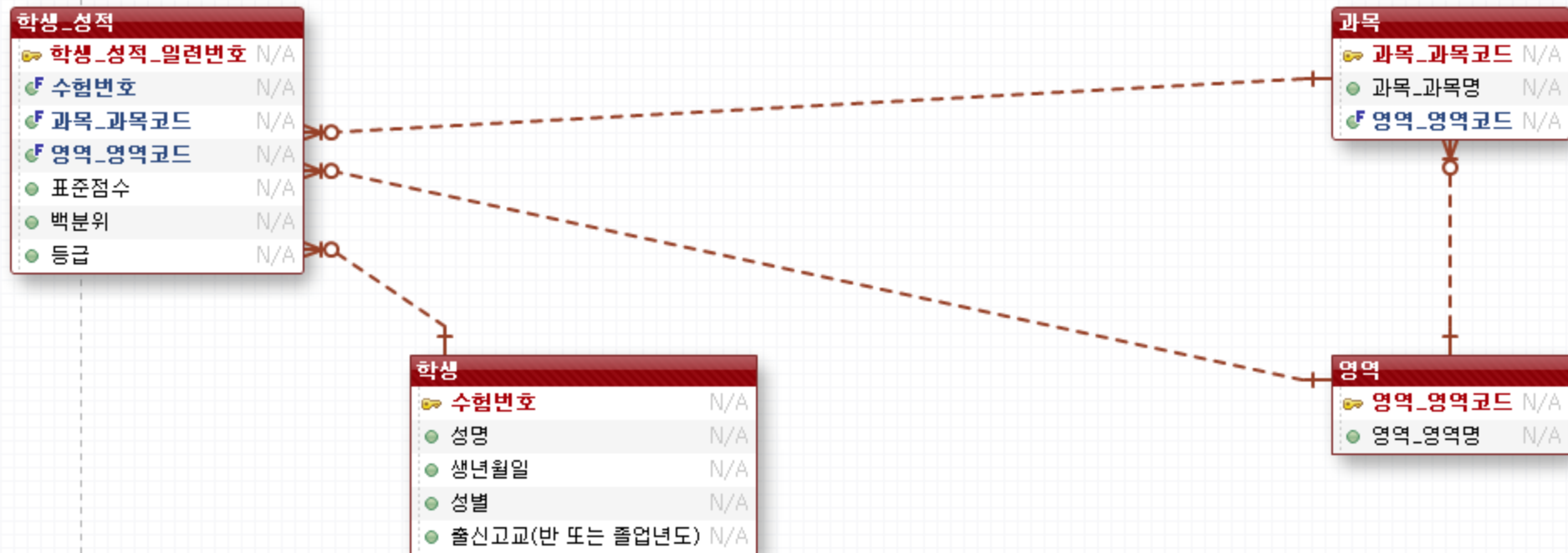


데이터모델링 미션#2

2022 수능성적표 3차 정규화 적용해 논리, 물리모델링하기

A014손은빈

1. 3차 정규화 - 영역명 속성을 분리함



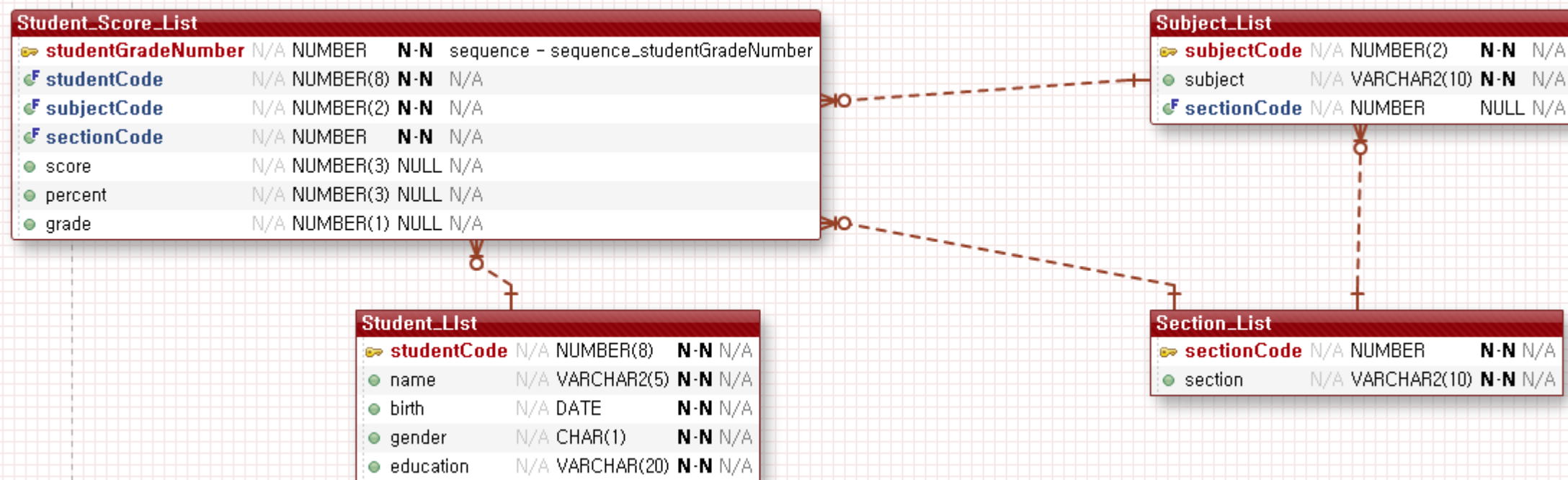
추가 논의사항

- 1) 학생 엔티티의 출신고교 속성을 신규 엔티티(가칭: 출신고교)로 생성 후 출신고교 속성을 이전시킬지 논의
- 2) 학생-성적 속성(영역-표준점수, 백분위, 등급의 접두사 변경 여부)
→ 앞에 접두사(영역-)삭제 결정
- 3) 표준점수 백분위 절대평가에 해당되는 영역 구분을 어떻게 할 것인가?
→ 표준점수 백분위의 NULL값 허용(단, DB or 프로그램 선에서 절대평가인 영역들을 처리하는 방향으로 논의 완료)

영역테이블 추가

- 영역테이블 추가, 영역명을 과목엔티티에서 옮김
- 장점1: 선택과목이 없는 영역의 점수를 표시함
- 장점2: 선택과목이 2개인 영역의 동일한 영역코드를 부여하고, 과목코드로 구분함
- 장점3: 2021 -> 2022로 영역란 수정시, 영역테이블에서 독립적으로 수정할 수 있음
- 단점1: 테이블 구조가 복잡해짐, 다수개 테이블을 조회해야 함.

2. 물리모델링 - 영문컬럼명, 데이터타입, 제약조건, 시퀀스 추가함



3. 샘플데이터 입력값 정하기

수험번호	성명	생년월일	성별	출신고교(반 또는 졸업연도)				
학생_수험번호	학생_성명	학생_생년월일	학생_성별	학생_출신고교(반 또는 졸업연도)				
20220001	홍길동	03.03.01	남	대한고등학교(1)				
영역	한국사영역	국어영역	수학영역	영어영역	탐구영역	탐구영역	제2외국어/한문영역	
선택과목	(엔티티_속성명)	과목_과목명	과목_과목명	과목_과목명	과목_과목명	과목_과목명	과목_과목명	과목_과목명
	입력값	null	화법과 작문	확률과 통계	null	생활과 윤리1	지구과학1	독일어1
표준점수	(엔티티_속성명)	학생_성적_표준점수	학생_성적_표준점수	학생_성적_표준점수	학생_성적_표준점수	학생_성적_표준점수	학생_성적_표준점수	학생_성적_표준점수
	입력값	null	133	114	null	64	62	null
백분위	(엔티티_속성명)	학생_성적_백분위	학생_성적_백분위	학생_성적_백분위	학생_성적_백분위	학생_성적_백분위	학생_성적_백분위	학생_성적_백분위
	입력값	null	98	69	null	91	87	null
등급	(엔티티_속성명)	학생_성적_등급	학생_성적_등급	학생_성적_등급	학생_성적_등급	학생_성적_등급	학생_성적_등급	학생_성적_등급
	입력값	1	1	4	1	2	3	1

과목	영역
10 한국사	10 한국사
20 화법과작문	20 국어
30 확률과통계	30 수학
40 영어	40 영어
50 생활과윤리1	50 탐구
51 지구과학1	50 탐구
60 독일어1	60 제2외국어/한문

4. DDL, DML시험하고 물리 모델링 수정하기

- 데이터 insert 오류 발생함
- > 데이터형 크기 수정함(성명, 출신고교, 과목명, 영역명)
- > 속성의 널체크 수정함(학생_성적 테이블의 과목코드를 NULL값 입력 가능하도록 함)

M_StudentScores		학생_성적	
물리 이름*	논리 이름*	도메인	데이터 타입
studentGradeNumber	학생_성적_일련번호	N/A	NUMBER N-N
studentCode	수험번호	N/A	NUMBER(8) N-N
subjectCode	과목_과목코드	N/A	NUMBER(2) NULL
sectionCode	영역_영역코드	N/A	NUMBER(2) N-N
score	표준점수	N/A	NUMBER(3) NULL
percent	백분위	N/A	NUMBER(3) NULL
grade	등급	N/A	NUMBER(1) NULL

M_Subjects		과목	
물리 이름*	논리 이름*	도메인	데이터 타입
subjectCode	과목_과목코드	N/A	NUMBER(2) N-N
subject	과목_과목명	N/A	VARCHAR2(45) N-N
sectionCode	영역_영역코드	N/A	NUMBER(2) N-N

M_Students		학생	
물리 이름*	논리 이름*	도메인	데이터 타입
studentCode	수험번호	N/A	NUMBER(8) N-N
name	성명	N/A	VARCHAR2(45) N-N
birth	생년월일	N/A	DATE N-N
gender	성별	N/A	CHAR(1) N-N
education	출신고교(반 또는 졸업년도)	N/A	VARCHAR2(200) N-N

M_Sections		영역	
물리 이름*	논리 이름*	도메인	데이터 타입
sectionCode	영역_영역코드	N/A	NUMBER(2) N-N
section	영역_영역명	N/A	VARCHAR2(45) N-N

추가 논의사항
 1) 학생 엔티티의 출신고교 속성을 신규 엔티티(가칭: 출신고교)로 생성 후 출신고교 속성을 이전시킬지 논의
 2) 학생_성적 속성(영역-표준점수, 백분위, 등급의 점두사 변경 여부)
 -> 앞에 점두사(영역)삭제 결정
 3) 표준점수 백분위 절대평가에 해당되는 영역 구분을 어떻게 할것인가?
 -> 표준점수 백분위의 NULL값 허용(단, DB or 프로그램 선에서 절대평가인 영역들을 처리하는 방향으로 논의 완료)

영역테이블 추가
 코드 속성 추가, 영역명을 과목엔티티에서 옮김
 장점1: 선택과목이 없는 영역의 점수를 표시함
 장점2: 선택과목이 2개인 영역의 동일한 영역코드를 부여하고, 과목코드로 구분함
 장점3: 2021 -> 2022로 영역란 수정시, 영역테이블에서 독립적으로 수정할 수 있음
 단점1: 테이블 구조가 복잡해짐, 다수개 테이블을 조회해야 함.

5. exERD 사용해 쿼리문 생성하기

- 이클립스

- > exERD

- > 포워드엔지니어링

포워드 엔지니어링

DDL 생성 옵션

Data Definition Language의 생성 옵션을 지정 해 주세요

설정: 마지막으로 설정 된 구성

공통

☐ 이름 앞에 스키마 표시 ☐ 이름을 따옴표(")로 감싸기

데이터베이스

☐ 테이블 스페이스 생성 ☐ 테이블 스페이스 삭제

시퀀스

☒ 시퀀스 생성 ☐ 시퀀스 삭제

평션 / 프로시저

☐ 평션 생성 ☐ 평션 삭제
☐ 프로시저 생성 ☐ 프로시저 삭제

트리거

☐ 트리거 생성 ☐ 트리거 삭제

뷰

☒ 뷰 생성 ☐ 뷰 삭제

테이블

☒ 테이블 생성 ☐ 테이블 삭제
☐ 물리적 특성들 (Storage 및 추가 속성)

인덱스

☒ PK Index 생성 ☐ PK Index 삭제
☒ FK Index 생성 ☐ FK Index 삭제
☒ Unique Index 생성 ☐ Unique Index 삭제
☒ Non-Unique Index 생성 ☐ Non-Unique Index 삭제
☐ 물리적 특성들 (Storage 및 추가 속성)

미리 보기:

```
1  /* sequence1 */
2  CREATE SEQUENCE sequence1
3      INCREMENT BY 1;
4
5  /* 내 테이블1 */
6  CREATE TABLE my_table1 (
7      my_pk1 INTEGER NOT NULL, /* 내 기본키 컬럼1
8      my_pk2 INTEGER NOT NULL, /* 내 기본키 컬럼2
9      my_column1 VARCHAR(20) NOT NULL, /* 내 컬럼1
10     my_column2 VARCHAR(20) /* 내 컬럼2 */
11 );
12
13 COMMENT ON TABLE my_table1 IS 'DDL 생성 예제 테이블'
14
15 COMMENT ON COLUMN my_table1.my_pk1 IS '내 기본키 컬럼1'
16
17 COMMENT ON COLUMN my_table1.my_pk2 IS '내 기본키 컬럼2'
18
19 COMMENT ON COLUMN my_table1.my_column1 IS '내 컬럼1'
20
21 COMMENT ON COLUMN my_table1.my_column2 IS '내 컬럼2'
22
23 CREATE UNIQUE INDEX my_table1_pk
24     ON my_table1 (
25         my_pk1 ASC,
26         my_pk2 ASC
27     )
```



< Back

Next >

Finish

Cancel

6. 테스트데이터 INSERT하고 확인하기

```
INSERT into
M_Sections(sectionCode , section)
VALUES(10, '한국사');
INSERT into
M_Sections(sectionCode , section)
VALUES(20, '국어');
INSERT into
M_Sections(sectionCode , section)
VALUES(30, '수학');
INSERT into
M_Sections(sectionCode , section)
VALUES(40, '영어');
INSERT into
M_Sections(sectionCode , section)
VALUES(50, '탐구');
INSERT into
M_Sections(sectionCode , section)
VALUES(60, '제2외국어/한문');
SELECT * FROM M_sections;
DELETE FROM M_sections;
```

SECTIONCODE	SECTION
1	10 한국사
2	20 국어
3	30 수학
4	40 영어
5	50 탐구
6	60 제2외국어/한문

```
INSERT into
M_Subjects(subjectCode , subject , sectionCode )
VALUES(10, '한국사', 10);
INSERT into
M_Subjects(subjectCode , subject , sectionCode )
VALUES(20, '화법과 작문', 20);
INSERT into
M_Subjects(subjectCode , subject , sectionCode )
VALUES(30, '확률과 통계', 30);
INSERT into
M_Subjects(subjectCode , subject , sectionCode )
VALUES(40, '영어', 40);
INSERT into
M_Subjects(subjectCode , subject , sectionCode )
VALUES(50, '생활과 윤리1', 50);
INSERT into
M_Subjects(subjectCode , subject , sectionCode )
VALUES(51, '지구과학1', 50);
INSERT into
M_Subjects(subjectCode , subject , sectionCode )
VALUES(60, '독일어1', 60);
SELECT * FROM M_subjects;
DELETE FROM M_subjects;
```

SUBJECTCODE	SUBJECT	SECTIONCODE
1	10 한국사	10
2	20 화법과 작문	20
3	30 확률과 통계	30
4	50 생활과 윤리1	50
5	51 지구과학1	50
6	60 독일어1	60
7	40 영어	40

6. 테스트데이터 INSERT하고 확인하기

```
INSERT into
M_StudentScores (studentGradeNumber , studentCode , subjectCode , sectionCode , score, percent , grade)
VALUES(sequence_studentGradeNumber.nextval, '20220001', NULL, 10, NULL, NULL, 1 );
INSERT into
M_StudentScores (studentGradeNumber , studentCode , subjectCode , sectionCode , score, percent , grade)
VALUES(sequence_studentGradeNumber.nextval, '20220001', 20, 20, 133, 98, 1 );
INSERT into
M_StudentScores (studentGradeNumber , studentCode , subjectCode , sectionCode , score, percent , grade)
VALUES(sequence_studentGradeNumber.nextval, '20220001', 30, 30, 114, 69, 4 );
INSERT into
M_StudentScores (studentGradeNumber , studentCode , subjectCode , sectionCode , score, percent , grade)
VALUES(sequence_studentGradeNumber.nextval, '20220001', NULL, 40, NULL, NULL, 1 );
INSERT into
M_StudentScores (studentGradeNumber , studentCode , subjectCode , sectionCode , score, percent , grade)
VALUES(sequence_studentGradeNumber.nextval, '20220001', 50, 50, 64, 91, 2 );
INSERT into
M_StudentScores (studentGradeNumber , studentCode , subjectCode , sectionCode , score, percent , grade)
VALUES(sequence_studentGradeNumber.nextval, '20220001', 51, 50, 62, 87, 3 );
INSERT into
M_StudentScores (studentGradeNumber , studentCode , subjectCode , sectionCode , score, percent , grade)
VALUES(sequence_studentGradeNumber.nextval, '20220001', NULL, 60, NULL, NULL, 1 );
DELETE FROM M_studentScores;
SELECT * FROM M_studentScores;
```

스크립트 출력 x 질의 결과 x

SQL | 인출된 모든 행: 7(0.001초)

	STUDENTGRADENUMBER	STUDENTCODE	SUBJECTCODE	SECTIONCODE	SCORE	PERCENT	GRADE
1	1	20220001	(null)	10	(null)	(null)	1
2	2	20220001	20	20	133	98	1
3	3	20220001	30	30	114	69	4
4	4	20220001	(null)	40	(null)	(null)	1
5	5	20220001	50	50	64	91	2
6	6	20220001	51	50	62	87	3
7	7	20220001	(null)	60	(null)	(null)	1

```
INSERT into
M_Students(studentCode, name, birth, gender, education)
VALUES(20220001, '홍길동', '03.03.01', '0', '대한고등학교(1)');
SELECT * FROM M_students;
```

스크립트 출력 x 질의 결과 x

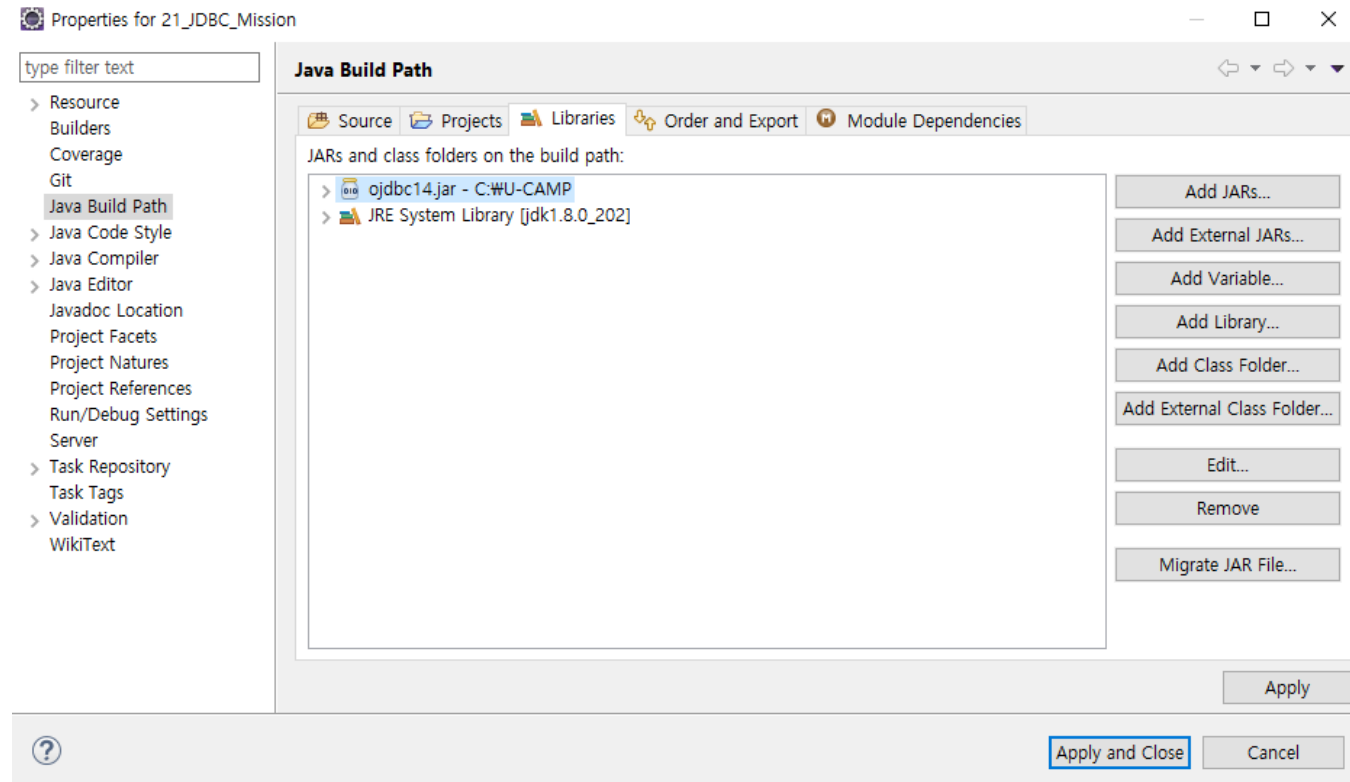
SQL | 인출된 모든 행: 1(0.001초)

	STUDENTCODE	BIRTH	GENDER	NAME	EDUCATION
1	20220001	03/03/01	0	홍길동	대한고등학교(1)

JDBC 미션#1

JDBC 5단계에 따라 SQL문 결과 확인하기
A014손은빈

1. JDBC 1단계 - 드라이버 로딩하기



```
// 1. driver는 각각 DBMS회사에서 구현 - 기본 제공 x, 수동으로 제공
Class.forName("oracle.jdbc.driver.OracleDriver");
System.out.println("1.driver loading OK");
```

2. JDBC 2단계 - 연결 생성하기

```
// 2. DB연결 서버의 정보 및 계정
String url = "jdbc:oracle:thin:@127.0.0.1:1521:XE";
String id = "hr";
String pw = "hr";
Connection conn=DriverManager.getConnection(url, id, pw);
System.out.println("2.DBMS 연결 OK");
```

드라이버 매니저가 다양한 DBMS로 접근
할 수 있도록 도움

3. JDBC 3단계 - 업무별 SQL작성하기

```
// 3. 4세트 ==> 업무에 따라 조회, 입력, 추가, 삭제, 수정
String sql =
    "SELECT\r\n" +
    "    e.employee_id\r\n" +
    "    , e.first_name || e.last_name\r\n" +
    "    , e.salary\r\n" +
    "    , d.department_name\r\n" +
    "FROM employees e, departments d\r\n" +
    "WHERE e.department_id = d.department_id\r\n" +
    "AND department_name = 'IT'\r\n";
Statement stmt=conn.createStatement();
ResultSet rs=stmt.executeQuery(sql);
```

Statement는 데이터베이스로 쿼리를 보내는
길이고, ResultSet은 데이터베이스로부터 결과
가 오는 길임

4. JDBC 4단계 - 결과 확인하기

```
// 4. 결과 확인하기
while(rs.next()) {
    System.out.println("end");
    System.out.println(rs.getInt(1)
        + " " + rs.getString(2)
        + " " + rs.getString(3)
        + " " + rs.getString(4));
}
```

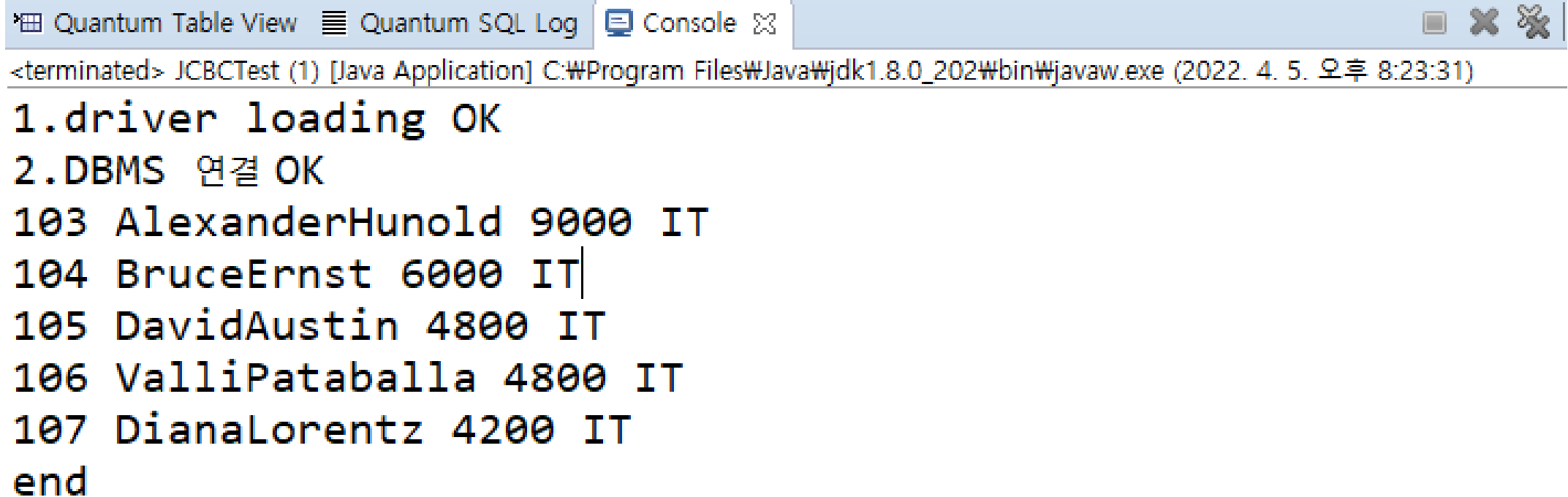
데이터베이스에 저장한 레코드의 자료형과 인덱스를 사용해 첫번째 행부터 마지막 행까지 결과를 가져옴

5. JDBC 5단계 - 자원 반환하기

```
//5. 서버는 공유 - 다쓰면 반환|  
rs.close(); stmt.close(); // 장바구니 - 일정 갯수만큼  
conn.close(); // 문 - 일정 갯수만큼  
System.out.println("end");
```

데이터베이스로의 연결과 쿼리전송, 결과값 저장을 위해 쓴 자원들은 공유자원이므로 반드시 반환해야 함

6. 실행 결과



```
<terminated> JCBCTest (1) [Java Application] C:\Program Files\Java\jdk1.8.0_202\bin\javaw.exe (2022. 4. 5. 오후 8:23:31)
1.driver loading OK
2.DBMS 연결 OK
103 AlexanderHunold 9000 IT
104 BruceErnst 6000 IT
105 DavidAustin 4800 IT
106 ValliPataballa 4800 IT
107 DianaLorentz 4200 IT
end
```

특정 부서(IT)의 사원번호, 이름, 급여를 확인함.
총 5명의 정보를 확인함.

