

DB 1/14 ~ 1/20

이것이 MySQL이다 책

MySQL 패스워드 : mcys1309

데이터베이스 학습 목표

자바 어플리케이션에서 DB에 접속하여 CRUD작업을 할 수 있다!

테이블을 작성할 수 있다. 1-N 관계의 두 테이블을 외래키로 연결할 수 있다.

*select쿼리 작성 중요

간단한 서브쿼리, 조인 쿼리 작성

집계함수 활용 + group by

조건절 활용. with and, or, like, in between~and

#1/14

책에서 우리가 필요한 부분만 찾아서 공부할 것.

점점 데이터가 몇십만개~~이상으로 필요한 프로그래밍이 생겨남. 데이터만 따로 저장할 공간이 필요해짐

DB파트가 여러개로 나뉘어짐

DB관리자, DB개발자, DB를 이용하는 앱 개발자

크게 이렇게 3파트가 있음. 우리는 이 중에서 DB를 이용하는 앱개발자에 속한다. 책의 학습 로드맵 맨 오른쪽에 있는 웹프로그래머, 응용프로그래머 부분 12,13,14,15

관리자는 2,3,5

DB개발자는 7,8,9,10,11

광범위하게 책을 다루고 있다.

우리의 포커스는 웹프로그래머! 대신 우리는 PHP,Python안쓰고 Java를 이용해 데이터를 연결하고 나중에 JSP/Servlet을 이용할 것이다.

데이터와 소통할 수 있는 언어. SQL

Structured Query Language.

데이터베이스 - 대용량의 데이터 집합을 체계적으로 구성해 놓은 것. 자바에서 배운 컬렉션 보다 대량의 개념. 여러명의 사용자가 동시에 접근이 가능해야 한다. 데이터의 저장 공간 자체!

우리가 자바에서 배운 컬렉션, File은 데이터를 영구적으로 보관하는 것. 프로그램을 종료해도 남아있음. 그전에 생성해놓은 데이터를 참조할 수 있다.

점점 대용량 데이터를 전문으로 다루는 것이 필요해짐. → 데이터베이스의 필요성

데이터베이스도 결국 file임. 자체적으로 알고리즘을 통해 정리작업을 할 수 있도록 만들어 놓은 것. 설계된 데이터. 정형화된 데이터. 데이터만을 가지고 고객이 요청한 문제를 해결하는 것이 데이터베이스! 체계화된 데이터를 이용해서! 모델링. 설계

설계된 데이터? 고객의 요청사항이 시작.

빅데이터는 비정형화된 데이터, 고객의 요청같은게 없음.

DBMS. 데이터베이스 매니지먼트 시스템. 데이터베이스를 관리 운영하는 역할. DB와 한 셋트로 보인된다. 유형은 계층형, 망형, 관계형, 객체지향형, 객체관계형 등이 있는데

SQL은 관계형 데이터베이스에서 사용되는 언어!

우리가 배울 것은 DBMS중 오라클 사의 MySQL(관계형)

DBMS, DB의 중요한 특징!

데이터의 무결성 : 데이터베이스 안의 데이터는 어떤 경로를 통해 들어왔던간에 데이터에 오류가 있어서는 안된다. 이를 위해 데이터베이스는 제약조건이라는 특성을 가진다.

데이터의 독립성 : DB의 크기를 변경하거나 데이터파일의 저장소를 변경해도 기존에 작성된 SW는 영향을 받지 않는다. 독립적인 관계
데이터 중복의 최소화 : 하나의 테이블에 저장하고 공유할 수 있다.
응용프로그램 제작 및 수정이 용이
데이터 안전성 향상 : 백업 복원기능 이용가능

SQL표준을 ANSI라고 함. 겹치는 부분들. 가장 기본적인CRUD관련부분들이 이에 속함. 표준이 계속 발전중.

표준 SQL을 기반으로해서 JDBC가 만들어짐.

나머지는 자바에서 인터페이스 추상클래스로 만들어놓고 각 데이터베이스 회사에서 오버라이드 하게끔하면 다른 프로토콜이어도 옮길 수 있다. 다른 회사로부터 뺏아오려면 자기네꺼에서 자바 언어로 된것을 오버라이드 해야함. 그래서 자바가 좋은 것.. 영향력이 있다.

MySQL은 오라클사에서 제작한 DMBS소프트웨어

드디어 데이터베이스 설치...!

오늘은 테이블에 대해서 설명하고 다음주에 직접 코드로 공부해볼 것.

데이터베이스 모델링-RDB(관계형)

현실세계에서 사용되는 데이터를 MySQL에 어떻게 옮겨놓을 것인지 결정하는 과정

SQL종류 여러개지만 다 같음

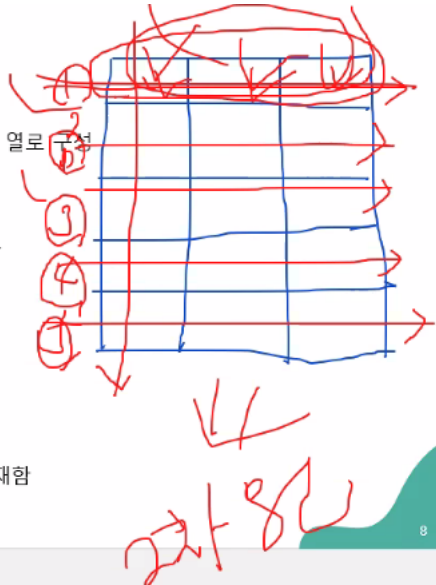
저장할 정보를 Table이라는 형식에 맞춰 저장한다.

데이터베이스 모델링과 필수 용어

- 데이터
 - 하나하나의 단편적인 정보
 - 정보는 있으나 아직 체계화 되지 못한 상태
- 테이블
 - 데이터를 입력하기 위해, 표 형태로 표현한 것
 - Ex) 회원 정보 테이블, 제품 정보 테이블
- 데이터베이스(DB)
 - 테이블이 저장되는 저장소
 - 각 데이터베이스는 서로 다른 고유한 이름을 가지고 있음
- DBMS (DataBase Management System)
 - 데이터베이스를 관리하는 시스템 또는 소프트웨어

데이터베이스 모델링과 필수 용어

- 열(=컬럼=필드)
 - 각 테이블은 열로 구성
 - 회원 테이블의 경우에는 아이디, 회원 이름, 주소 등 3개의 열로 구성
- 열 이름
 - 각 열을 구분하기 위한 이름
 - 열 이름은 각 테이블 내에서는 중복되지 않고, 고유해야 함
- 데이터 형식
 - 열의 데이터 형식
 - 테이블을 생성할 때 열 이름과 함께 지정
- 행(=로우=레코드)
 - 실질적인 데이터
 - 회원 테이블의 경우 4건의 행 데이터, 즉 4명의 회원이 존재함



>> 이것이 MySQL이다

이런 5행 3열 테이블에서

'대상'은 5개. 5개의 대상의 의미는 같음. 여기서 대상이란 뭐냐 Entity. 테이블은 이 엔티티에 대한 정보를 가지고 있다.

ex) 대상 = Entity : 상품

테이블안에는 상품의 속성들이 들어감

행 = 로우 = 레코드 를 튜플이라고도 함.

열 이름 다음에 데이터형식이 나와야하는데 열의 데이터 형식을 지정해줘야함

열 이름과, 데이터 형식을 합쳐서 **스키마**라고함

↳ 유튜브 강의에서는 데이터베이스랑 스키마랑 같은 개념으로 생각하면 된다고 했는데. 모르겠다. 음 둘다 맞는듯. **스키마는 데이터베이스의 테이블 구조 및 형식, 관계 등의 정보를 형식 언어로 기술한 것 = 데이터베이스**

테이블을 만들 때 테이블을 만드는 약속

각 레코드는 최소한 한개의 열은 다른 레코드와 구분이 되어야 한다. 모든 정보가 같으면 안됨. 테이블에서 레코드는 중복 되어선 안된다!

중복되지 않는 고유한 값을 가지는 속성을 기본키 Primary Key로 설정할 수 있다. 각 행을 구분하는 유일한 열!

- **기본 키 (Primary Key) 열**
 - 기본 키(또는 주 키) 열은 각 행을 구분하는 유일한 열
 - 중복되어서는 안되며, 비어 있어서는 안 됨
 - 각 테이블에는 기본 키가 하나만 지정
- **외래 키(Foreign Key) 필드**
 - 두 테이블의 관계를 맺어주는 키
 - 4장 이후 설명
- **SQL (Structured Query Language)**
 - 구조화된 질의 언어
 - 사람과 DBMS가 소통하기 위한 말(언어)
 - 6, 7장에서 자세히 다룸

인공 기본키가 사용될 때도 있는데 데이터가 중복될 수 있는 것들일 경우 sequence를 이용해서 순서를 지정해주는 등의 방식을 사용함. 이를 인공 기본키라고 한다. 그냥 기본키로 생각하면 됨! auto_increment

스키마 (Schema) 생성

- MySQL에서는 스키마와 데이터베이스가 완전히 동일한 용어로 사용
- Workbench의 [SCHEMAS]의 빈 부분
 - 마우스 오른쪽 버튼 클릭 후 [Create Schema](=Create Database) 선택
 - CREATE SCHEMA 'shopdb'문을 쿼리 창에서 입력하는 것과 동일한 작동
 - 이름 입력하면 DB 생성
- 왼쪽 데이터베이스 목록에 shopdb 데이터베이스 확인
- 아무것도 들어있지 않은 데이터베이스 생성

51p 56p비밀번호 집어넣는 거, 58p

60 ~ 71p까지 실습하면 된다. 테이블생성, 데이터 집어넣기(GUI로)

인덱스와 뷰는 실습말고 읽기만

스토어드프로시저,트리거,데이터베이스백업및관리 는 패스

80~105p는 안봐도된다. 93p

#1/17

Ch4. 데이터베이스 모델링

프로젝트 - 현실세계의 업무를 컴퓨터 시스템으로 옮겨놓는 일련의 과정. 대규모의 프로그램을 작성하기 위한 전체 과정.

과거에 비해 복잡한 작업이 요구되는 프로젝트가 많아짐, 프로젝트의 규모가 커지면서 무턱대고 코딩을 바로 시작하는 것이 아닌 분석과 설계 작업에 보다 시간을 투자할 필요가 생김 → 소프트웨어 개발 방법론 등장

전통적으로 사용되어 왔던 방법론은 폭포수 모델이었음. 각 단계가 끝나면 바로 다음단계로 넘어가는 모델. 프로젝트의 진행 단계가 명확해지는 장점이 있으나 문제가 발생했을 때 앞 단계로 다시 거슬러 올라가기 어렵다는 문제가 있다.

업무 분석과 시스템 설계가 중요!!

데이터베이스 모델링 - 현실세계에서 사용되는 작업이나 사물들을 DBMS의 데이터베이스 개체로 옮기기 위한 과정. → 현실에서 쓰이는 것을 테이블로 변경하기 위한 작업

테이블로 옮기기 직전 상태까지 만드는 것 121p의 상태!! **모델링을 통해 고객 테이블과 구매 테이블을 1:N, PK FK 관계를 맺어주었고(제약 조건이 자동으로 설정) 열이름과 데이터 형식, NULL여부 결정 하는 테이블 구조를 정의 함!!**

115p~ 쇼핑몰 데이터 베이스 모델링 예제 책볼것.

고객 테이블과 구매 테이블이 구분됨. 근데 둘은 밀접한 관계가 있는 테이블 → 두 테이블의 업무적인 **연관성**을 맺어줘야함 → 이를 관계라고 한다! Relation

두 테이블 중 주(Master)가 되는 쪽을 부모, 상세(Detail)가 되는 쪽을 자식으로 설정한다. 보통 기준이 하나인 것을 부모, 기준이 여러개의 기록을 남기는 것을 자식으로 구분. 고객, 구매 테이블을 예로 보면 고객 한명이 여러개의 물건을 구매할 수 있으니 고객이 부모, 구매가 자식.

이러한 관계를 테이블의 1대다(1:N) 관계라고 지칭하고 관계형 데이터베이스에서 가장 보편적인 테이블 사이의 관계이다.

관계를 맺어주는 역할은 기본키와 외래키 설정을 통해 함. 고객테이블에서는 기본키를 고객 이름으로 설정. 자식 테이블의 외래 키는 부모 테이블의 기본 키와 일치되는 구매 테이블의 고객 이름으로 설정.

외래 키?! - 외래 키를 가지고 부모 테이블로 찾아가면 유일하게 하나의 정보를 얻을 수 있다!

테이블 생성의 기본 개념을 배워왔다!

4챕터 끝.

3장에서 워크벤치 GUI - 클릭클릭으로 테이블 만들어 봤던걸 이제 SQL코드로 만들어 볼 것임. 문법이 필요하다~

Ch8.

SQL로 테이블 생성! 195p참고. 실습2

Ch6. 기본 문법. 자주사용하는 구문

먼저 select. 가장 자주 사용하는 구문

select 열이름 from 테이블이름 - 해당 열 조회

열이름에 *작성하면 모든 컬럼인데 테이블에 있는 순서 그대로 적용됨. - 전체조회

여러개의 열을 가져오고 싶을 때는 콤마로 구분.

열 이름 각각을 **as**를 통해 별칭으로 지정하면 조회할 때 별칭으로 열이름이 나온다.

열 이름의 순서는 출력하고 싶은 순서대로 배열 가능!! 테이블 순서랑 다르게 바꿔서 출력할 수 있다.

특정 조건의 데이터만 조회하고 싶을 때는 **where**키워드를 추가해서

select...from...where... 처럼 사용.

select * from usertbl where name = '김경호' 조건에 맞는 값을 다 가져온다

관계연산자 **or, and, not**와 조건 연산자조합해서 조건에 사용할 수 있다. 여러 조건을 줄 수 있음. **주의할 것은 SQL조의 건연산자에서는 등호를 =한개밖에 안쓴다.**

- 내가 지금 하고자 하는 것. 대상이 생각이 안날 때. 예를 들어서 데이터베이스 이름, 테이블 이름, 필드 이름이 정확히 기억 안날때는?

1. **SHOW DATABASES;** (데이터베이스 목록들 보여준다)
2. **USE employees;** (데이터베이스는 이건거 같다. 찾았다! 다음으로 테이블은 뭐였더라)
3. **SHOW TABLE STATUS;** 현재 데이터 베이스의 **테이블**을 모두보여줌
4. **DESCRIBE employees;** employees라는 테이블의 열을 모두 보여줌. 테이블의 정보
5. **SELECT first_name, gender FROM employees;** 보여준 열에서 원하는 열정보를 조회한다.

SECTION 01 SELECT문

특정 조건의 데이터만 조회 - **<SELECT ... FROM ... WHERE>**

- **BETWEEN... AND**와 **IN()** 그리고 **LIKE**

- 데이터가 숫자로 구성되어 있으며 연속적인 값 : **BETWEEN ... AND** 사용

• ex)

```
SELECT name, height FROM usertbl WHERE height BETWEEN 180 AND 183;
```

- 이산적인(Discrete) 값의 조건 : **IN()** 사용

• ex)

```
SELECT name, addr FROM usertbl WHERE addr IN ('경남', '전남', '경북');
```

- 문자열의 내용 검색 : **LIKE** 사용(문자뒤에 % - 무엇이든 허용, 한 글자와 매치 '_' 사용)

• ex)

```
SELECT name, height FROM usertbl WHERE name LIKE '김%';
```

>> 이것이 MySQL이다

12

연속적인 값을 가지는 숫자는 비트윈 앤드 사용가능(이때 해당 숫자까지 포함 =) 하지만 문자열의 경우 연속적인 데이터가 아님. 이산적인 값 → **in()** 사용

문자열의 내용 검색 중요! **LIKE** '김%' 에서 %김으로 시작하는 어떤 글자든 검색하는 것.

는 한글자와 매치. **''**종신' 하면 맨앞 글자 아무거나 한글자이고 다음이 종신인 사람

ANY/ALL/SOME ,서브쿼리(SubQuery, 하위쿼리)

서브쿼리

- 쿼리문 안에 또 쿼리문이 들어 있는 것
- 서브쿼리 사용하는 쿼리로 변환 예제
 - ex) 김경호보다 키가 크거나 같은 사람의 이름과 키 출력
 - WHERE 조건에 김경호의 키를 직접 써주는 것을 쿼리로 해결

```
SELECT name, height FROM usertbl WHERE height > 177;
```



```
SELECT name, height FROM usertbl  
WHERE height > (SELECT height FROM usertbl WHERE Name = '김경호');
```

- 서브쿼리의 결과가 둘 이상이 되면 에러 발생

데이터가 많을 때 그 값을 다 기억하기 어렵고, 숫자를 직접 입력하는 것은 하드코딩이기 때문에 해당 조건을 다시 select문을 이용해 사용하는 것. 177이 결국 김경호씨의 키니까

*하위쿼리가 둘 이상의 값을 반환하는 경우에는 에러가 발생하니 주의!

SECTION 01 SELECT문

ANY/ALL/SOME ,서브쿼리(SubQuery, 하위쿼리)

ANY

- 서브쿼리의 여러 개의 결과 중 한 가지만 만족해도 가능
- SOME은 ANY와 동일한 의미로 사용
- '= ANY(서브쿼리)'는 'IN(서브쿼리)'와 동일한 의미



ALL

- 서브쿼리의 결과 중 여러 개의 결과를 모두 만족해야 함

이러한 문제를 해결할 수 있는 방법이 ANY구문. 해당 조건 중 하나라도 충족하면 출력.

ALL은 모든 조건을 만족시키는 것을 출력.

=ANY의 경우 각 조건에 해당하는 경우 출력이니까 결국 IN(서브쿼리)와 같다.

SECTION 01 SELECT문

원하는 순서대로 정렬하여 출력 : ORDER BY

◦ ORDER BY절

- 결과물에 대해 영향을 미치지 않는 출력되는 순서를 조절하는 구문
- 기본적으로 오름차순 (ASCENDING) 정렬
- 내림차순 (DESCENDING)으로 정렬하려면 열 이름 뒤에 DESC
- ORDER BY 구문을 혼합해 사용하는 구문도 가능
 - 키가 큰 순서로 정렬하되 만약 키가 같을 경우 이름 순으로 정렬

```
SELECT name, height FROM usertbl ORDER BY height DESC, name ASC;
```

- ASC(오름차순)는 디폴트 값이므로 생략 가능

결과물에 영향X. 출력순서만 조절. 오름차순이 기본이고 내림은 DESC를 붙여주면된다.

보통 select구문에서 가장 마지막에 사용한다고 생각하면 될듯

- 중복된 것은 하나만 남기는 DISTINCT
 - 중복된 것을 골라서 세기 어려울 때 사용하는 구문
 - 테이블의 크기가 클수록 효율적
 - 중복된 것은 1개씩만 보여주면서 출력
- 출력하는 개수를 제한하는 LIMIT
 - 일부를 보기 위해 여러 건의 데이터를 출력하는 부담 줄임
 - 상위의 N개만 출력하는 LIMIT N 구문 사용
 - 개수의 문제보다는 MySQL의 부담을 많이 줄여주는 방법
- 테이블을 복사하는 CREATE TABLE ... SELECT
 - 테이블을 복사해서 사용할 경우 주로 사용
 - CREATE TABLE 새로운 테이블 (SELECT 복사할 열 FROM 기존테이블)
 - 지정한 일부 열만 복사하는 것도 가능
 - PK나 FK 같은 제약 조건은 복사되지 않음

SELECT DISTINCT addr FROM usertbl; 처럼 distinct 키워드를 원하는 열이름 앞에 붙여주면 중복을 제거한 결과를 출력.

order by 정렬기준 LIMIT 출력개수; 처럼 사용하면 출력개수만큼만 출력. 원하는 만큼만 출력개수를 조절할 수 있다. 자주사용!

기존 DB 데이터 테이블을 건드리지는 건 매우 위험한 일이니까 연습용으로 사용하려고 테이블을 복사해오는 건데 굳이 몰라도 된다. 제약조건은 복사되지 않는다.

6.1.3 GROUP BY 및 HAVING 그리고 집계 함수

먼저 GROUP BY는 그룹으로 묶어주는 역할. 각 유저가 여러 종류의 물건을 샀을 때 산 물건의 총 개수를 구하려면 각각의 물건의 amount 값을 더해야 하는데 이를 유저 아이디로 그룹화하면 사용자별로 해당 값을 합쳐서 출력하게 된다.

```
SELECT userID, SUM(amount) FROM buytbl GROUP BY userID;
```

가장 큰 키와 가장 작은 키가진 사람 출력하는 코드. 책 215p 6-38

select name, max(height), min(height) from usertbl; 만으로는 이름이 하나밖에 안나와서 큰키와 작은키중 누구것인지 모름.

```
select name, height from usertbl
where height in (select max(height) from usertbl) or height in (select min(height) from usertbl);
```

```
select name, height from usertbl
where height = (select max(height) from usertbl)
or height = (select min(height) from usertbl)
order by name;
```

아래 코드가 교재 코드고 위예가 내가 같은 코드로 바꿔본거. 아까 ANY가 in이랑 같다고 했던게 생각나서 any나 or이나 둘다 하나만 만족하면 되는거라고 생각해서 or도 in으로 표현할 수 있지 않을까 해서 = 대신 in으로 바꿔봄

Having절

group by는 집계함수 sum max min같은 것들과 자주 함께 쓰이는데 집계함수에 대해 조건에 맞는 것들만 출력하고 싶을때는?!

우리가배운 조건은 where문인데 집계함수는 where절에 나타날 수 없다! 오류.

이때 사용하는 것이 HAVING절. 집계함수에 대해 조건을 제한하는 용도.

***반드시 group by절 다음에 나와야 한다. 순서 주의!! group에 대한 조건을 사용할 때**

추가로 오름차순, 내림차순 사용하려면 마지막에 order by 기준 사용

with rollup문

group by절과함께 사용해서 분류별로 합계 및 총합을 구할 수 있다.

```
형식 :
SELECT select_expr
FROM table_references
WHERE where_condition
GROUP BY {col_name | expr | position}
HAVING where_condition
ORDER BY {col_name | expr | position}
```

이건 외울것!! 가장 중요하다. 순서.

6.1.4 SQL의 분류

1. DML - 데이터 조작 언어. manipulation. CRUD에 사용되는 언어. DML구문이 사용되는 대상은 테이블의 행! 따라서 DML을 사용하기 이전에 반드시 테이블이 정의되어 있어야 한다. 트랜잭션(실제 테이블에 완전히 적용하지 않고 임시로 적용하는 것, 취소가능)이 발생하는 SQL이다. (선택트는 DQL로 보기도함. 테이블에 영향을 미치지 않아서)
2. DDL - 데이터 정의 언어. definition. 데이터베이스, 테이블, 뷰, 인덱스 등의 데이터베이스 개체를 생성/삭제/변경하는 역할. CREATE,DROP,ALTER가 대표적. *트랜잭션을 발생시키지 않는다. 실행즉시 MYSQL에 적용!
3. DCL - 데이터 제어 언어. control 사용자에게 권한을 부여하거나 빼앗을 때. GRANT/REVOKE/DENY

6.2 데이터의 변경을 위한 SQL문

CRUD관한거. 선택트는 조회로 이미 배웠고 변경에 관한 CUD!! 매우중요

- INSERT의 기본 형식

INSERT [INTO] 테이블[(열.....(가변))] VALUES (값...(가변));

테이블 뒤에 열을 생략하면 values안의 순서 및 개수가 테이블 정의된 열 순서 및 개수와 같아야 한다!

내가 입력하고 싶은 열만 하고 싶으면 테이블 안에 넣으면 됨. 나머지 열목록에는 NULL이 들어간다.

insert into를 매 데이터마다 입력하지 않고 한문장으로도 가능

INSERT INTO 테이블이름 VALUES

(, , ,), (, , ,), (, , ,), (, , ,);

이런식으로 값들만 이어서 입력

- UPDATE의 기본 형식

UPDATE 테이블이름

SET 열1=값1, 열2=값2 ...

WHERE 조건;

***WHERE절은 생략가능 하지만 생략하면 테이블의 모든 행이 변경된다**

ex.

UPDATE testTbl4

SET Lname = '없음'

WHERE Fname = 'Kyoichi';

교이치의 라스트네임을 없음으로 변경!

- DELETE의 기본형식

DELETE FROM 테이블이름

WHERE 조건;

UPDATE와 거의 비슷한 개념. **딜리트는 행단위로 삭제한다.**

마찬가지로 where 생략되면 전체 데이터 삭제

조건에 해당하는 모든 행을 삭제하지만 LIMIT을 통해 상위 N개만 삭제지정할 수 있다.

이제 Ch8. 테이블과 뷰

8.1 테이블

제약조건 및 테이블의 수정에 대해 알아보자!

8.1.2 제약 조건

제약조건이란 데이터의 무결성을 지키기 위한 제한된 조건을 의미. 특정 데이터를 입력할 때 무조건적으로 입력되는 것이 아니라 어떤 조건을 만족했을 때 입력되도록 제약

어떤 사이트에서 같은 아이디로 회원가입하면 생성불가능한 것이 예시. 아이디가 기본키니까! 동일한 것이 들어갈 수 없는 제약조건이 있어서 그렇다.

MySQL에서는 데이터의 무결성을 위한 6가지의 제약 조건이 있다.

- 기본키 제약조건 - 기본 키에 입력되는 값은 중복될 수 없으며 NULL값이 될 수 없다.
- 외래키 제약조건 - 외래 키 테이블에 데이터를 입력할 때에는 기존 테이블에 이미 데이터가 존재해야 한다.
- UNIQUE 제약조건 - 중복되지 않는 유일한 값을 입력해야하는 조건
- CHECK 제약조건
- DEFAULT 정의
- NULL값 허용

테이블과 뷰 개념 간단하게 까지 진도나갔다. 내일은 고급쿼리, 함수

#1/18

어제 배운 내용 예제문제 풀이! 중요!!

```
use shopdb;

-- buytbl에서 groupName이 지정되지 않는 물품을 산 userid를 출력하시오
select distinct userid from buytbl where groupname is null;

-- buytbl에서 groupName이 지정되지 않는 물품을 산 user의 정보를 출력하시오
```

```

select * from usertbl
where userid in (select userid from buytbl where groupname is null);

-- EJW가 산 물품의 목록과 반복구매한 횟수를 출력 (구매수량 말고 구매 몇번 했냐)
select prodname, count(*)
from buytbl where userid = 'ejw' group by prodname;

-- user가 구매한 목록을 중복없이 출력
select distinct prodname from buytbl;

-- user가 구매한 물품중 가장 비싼 상품과 싼 상품에 대한 정보 출력
select distinct prodname, groupname, price from buytbl
where price = (select max(price) from buytbl)
or price = (select min(price) from buytbl);

-- user의 이름을 오름차순으로 정렬하시오
select name from usertbl order by name;

-- 휴대전화1,2를 모두 가지고 있는 사람은 몇명인가?
select count(*) as '휴대폰 있는 사람'
from usertbl
where mobile1 is not null and mobile2 is not null;

-- userid에 S가 있는 회원의 정보를 출력하시오
select * from usertbl where userid like '%S%';

-- userid에 S가 있는 회원중 나이가 가장 많은 사람과 작은 사람의 나이를 출력하시오
select max(A.birthyear)-min(A.birthyear) as 나이차
from (select * from usertbl where userid like '%S%') as A;

-- 가장 많은 쇼핑 횟수를 가진 유저는
select userid, count(userid)
from buytbl
group by userid
order by count(userid) desc limit 1;

```

이제 Ch.7 SQL 고급진도

처음 배울 때 변수가 따로 없어서 조건식에서 =한개만 사용한다고 했는데 변수도 있고 함수도 있고 프로그래밍 파트도 있는데 이걸 뭐지?

DB개발자들을 위한 것들임

데이터 형식은 보고, 내장함수들은 체크한것만 간단히 보면될듯. 문자열함수는 우리가 쓸일 거의 없다. 제어 흐름 함수는 중요한 CASE~WHEN~ELSE~END는 중요! CASE는 내장함수는 아니고 연산자로 분류되는데 자바의 switch문과 비슷하니 알아둘것!

중요한건

7.2 조인 ***

두 개 이상의 테이블을 서로 묶어서 하나의 결과 집합으로 만들어 내는 것

종류 : INNER JOIN, OUTER JOIN, ~~CROSS JOIN~~, SELF JOIN

아래는 조인을 이해하기 위해 꼭 필요한 개념! 다시 한번 생각해보자.

#데이터베이스의 테이블

- 중복과 공간 낭비를 피하고 데이터의 무결성을 위해서 여러 개의 테이블로 분리하여 저장
- 분리된 테이블은 서로 관계를 가짐
- 1대 다 관계가 보편적

1. INNER JOIN 내부조인

- 조인 중에서 가장 많이 사용되는 조인
- 일반적으로 join이라고 얘기하는 것이 이 INNER JOIN을 지칭
- 사용 형식

- 사용 형식

```
SELECT <열 목록>  
FROM <첫 번째 테이블>  
INNER JOIN <두 번째 테이블>  
ON <조인될 조건>  
[WHERE 검색조건]
```

```
use shopdb;  
select *  
from buytbl  
inner join usertbl  
on buytbl.userid = usertbl.userid  
where buytbl.userid = 'jyp';
```

2. OUTER JOIN 외부조인

- 조인의 조건에 만족되지 않는 행까지도 포함시키는 것
- LEFT OUTER JOIN - 왼쪽 테이블의 것은 모두 출력되어야 한다면 이해하면 됨. LEFT JOIN으로 줄여서 쓸 수 있다
- RIGHT OUTER JOIN - 오른쪽 테이블의 것은 모두 출력되어야 한다.

```
-- 외부조인  
select u.userid, u.name, b.prodName  
from usertbl U  
left outer join buytbl B  
on u.userid = b.userid;
```

3. CROSS JOIN 상호 조인


- 한쪽 테이블의 모든 행들과 다른 쪽 테이블의 모든 행을 조인시키는 기능.
- 크로스조인의 결과개수 = 두 테이블 개수를 곱한 개수 (카티전 곱)
- 위에서 내부 외부 조인은 ON을 통해 조건을 정했는데 상호조인은 그거 없이 그냥 모든 내용 조인하는 것.

4. SELF JOIN 자체 조인

- 따로 개념이 있는 것이 아님. 별도의 구문은 없음
- 자기 자신과 자신이 조인하는 것.
- 결국 안에서 내부조인 하는 것

```
create table employees(id int primary key, name varchar(10), mgr_id int);  
insert into employees values (1, '김사장', NULL);  
insert into employees values (2, '이전무', 1);  
insert into employees values (3, '박부장', 2);  
insert into employees values (4, '최과장', 3);  
insert into employees values (5, '박대리', 4);  
  
-- 셀프 조인  
select * from employees e  
[inner] join employees m  
on e.mgr_id = m.id;
```

Visual JOIN

 <https://joins.spathon.com/>

JOIN개념 참고 사이트!!

이제 JDBC프로그래밍 할거임. 자바와 DB연결! +서버

이클립스 perspective를 EE로 변경!

데이터 소스 익스플로러에서 new해가지고 mysql , 내 shopdb를 연동시켰다!

이제 코드로 연동시키는 방법. 이걸 거의 외워야함. 자동으로 해야하는 코드

```
package mc.js.lessonB;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class MysqlMain {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MysqlMain mm = new MysqlMain();
        try {
            mm.test();
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public void test() throws ClassNotFoundException, SQLException {
        //서버 접속에 필요한 정보
        String url = "jdbc:mysql://localhost:3306/shopdb";
        //접속한 뒤에 작업을 하기 위해 필요한 라이브러리 클래스
        String driver = "com.mysql.jdbc.Driver"; //클래스임
        //아이디 및 비번
        String id = "root";
        String pwd = "mcys1309";
        //드라이버 매니저라는 곳에서 위의 url에접속해서 아이디 비번 가지고 접속
        //프로토콜 간의 약속이 있어야 상호간의 데이터 처리가능하데
        //mysql이 만든 드라이버가 보증해주는거. 약속OK - mysql파일받아서 활동OK해줌
        Class.forName(driver); // 로딩코드
        Connection con = DriverManager.getConnection(url,id,pwd);
        //커넥션을받아와서 이제 CRUD작업을 할것임 -> 테이블필요
        if(con!=null) {
            System.out.println("connected");
            con.close();
        } else {
            System.out.println("fail connection");
        }
    }
}
```

▼ DB코드 이해!

@\$\$\$@ -db연동 이해- @\$\$\$@

DB와 연동하는법 : JDBC가 필요함.

JDBC (자바 데이터베이스 커넥티비티) 란?

JAVA의 CLASS중 하나로 DB와의 연결을 가능케 함.

JDBC 구동시 필요한것 : JDBC 드라이버 (jar파일 내장 클래스)

JDBC 드라이버 구동시 필요한것 : jar파일 라이브러리 적용 및 로딩코드 입력

『

DB별 로딩코드

MySQL : Class.forName(**com.mysql.jdbc.Driver**)

ORACLE : Class.forName(oracle.jdbc.driver.OracleDriver)

MSSQL : Class.forName(com.microsoft.sqlserver.jdbc.SQLServerDriver)

』

※현재 우리 과정에서 'mysql-connector-java-8.0.26.jar' 내의 Driver 경로는

※ 'com.mysql.jdbc.Driver' 이다.

※따라서 로딩코드 사용시 'Class.forName(com.mysql.jdbc.Driver)'을 사용해야한다.

『

JDBC의 클래스 (java.sql.*)

Connection : DB와 연결해주는 하나의 클래스

Statement : 연결된 DB의 쿼리를 실행하는 클래스

ResultSet : 쿼리의 결과를 가져오는 클래스

』

이제 이클립스와 디비가 연결된 상태. 공유되어있음. CRUD 작업할 것. 아직 이클립스에서 코드작성이 익숙치 않으니 워크벤치에서 만들어도 됨.

gisatable을 만들어볼거임. 자바코드-이클립스에서 데이터베이스로 전송할 때 하는 방법 배우는 거

가장먼저해야할일이 뭐까?

DB에 의사소통을 전달해야하는데 SQL을 통해해야한다. 쿼리가 필요!!

쿼리문 작성 → 스트링 버퍼나 빌더 사용

스트링 버퍼와 빌더의 차이는 동기화 여부.

버퍼가 동기화되어있어서 멀티쓰레드에 안전함 데이터 보호!

싱글쓰레드프로그램의 경우 동기화는 불필요한 성능저하가 될수있지만 거의 멀티니까 신경안쓰고 버퍼쓰면될듯

@2022년 1월 19일

#1/19

이클립스 basic.sql스크립트에서 쿼리문 작성. dmp, dept 테이블!

기억할 것은 emp_hiredate에 있는 now() 함수. 타임이 datetime이니까 연월시분초가 들어감

```
drop table dept;
create table dept(
    dept_code char(5) not null primary key,
    dept_name varchar(20) not null,
    dept_loc varchar(20) not null
);
insert into dept values
('ABCDE', '개별', '강남'), ('BCDEF', '설계', '강북'), ('CDEFG', '분석', '강서');
select * from dept;

drop table emp;
create table emp(
    emp_code int auto_increment primary key,
    dept_code char(5) not null,
    emp_name varchar(10) not null,
    emp_age int not null,
    emp_hiredate datetime not null default now(),
    emp_salary int not null default 3000,
    foreign key(dept_code) references dept(dept_code)
);
insert into emp values(null, 'CDEFG', 'KIM', 30, default, default);
insert into emp values(null, 'ABCDE', 'LEE', 20, default, 5000);
-- 디폴트로 설정된 칼럼에 값을 입력하면 입력한 값으로 생성
insert into emp(dept_code, emp_name, emp_age) values('BCDEF', 'PARK', 25);
-- 테이블안의 열을 설정해서 원하는 열에만 값을 입력할 수 있다. 입력안한 열은 기본값, 없으면 안들어감
-- 대신 not null로 설정된 열이 있어 있다면 오류발생
-- 중복되면 안되는 PK가 1씩 증가하게끔 설정되어 있기 때문에 나머지 컬럼에 같은데 데이터를 입력해도 오류 X
select * from emp;
```

이렇게 엑셀파일 - 실패테이블 정의서 참고해서 테이블 만들고 삽입하는 과정 복잡함

이제 어제 이클립스 sql연동하던거 이어서JDBC설명! JDBC 튜토리얼 pdf p27~참고!

커넥션을 얻기위해서 import하고 커넥션.

url주소로 가서 DB한테 아이디 비번주고 요청. DB는 난 뭐하는 애냐 하다가 아이디 비번 불러와서 맞는거 보고 해당 ID에 맞는 권한만큼 데이터를 넘겨줄 수 있음.

가장먼저 해야할 것은 JDBC드라이버를 등록하는것.

첫번째 방법

Class.forName() - 가장 많이 사용. 자동으로 가져온다. 혹시나 인스턴스가 생성되지 않을 것을 걱정해서 명시적으로 작성할 수 있음.

Class.forName().newInstance(); 그냥 이런 방법이 있구나

인스턴스가 생성된다는 것을 명시적으로 표시하고 싶을때!

드라이버매니커 젓코넥션메소드를 통해 커넥션을 확정. 확립함.

getConnection(String url, String user, String password)

3종류 중에 우리가 사용하는 코드!

DriverManager.getConnection(url,id,pwd);

jdbc:mysql://hostname/ databaseName

여기서 jdbc:mysql: 이 앞부분이 프로토콜. 약속

이제 우리가 어제 작성한 파일로 가서 기사테이블 쿼리를 작성해볼 거임

입력할 데이터가 너무 많아서 한줄씩작성하기위해 스트링버퍼를 사용한다고 했다!

스트링버퍼를 가지고 sql쿼리를 작성했고 이걸해보니 상당히 복잡하더라. 애를 어떻게 효율적으로 바꿀것인가?! 오후시간에 해결

현재상태는 insert까지 한 상태 test1 - insert

맨 아래 test()메소드는 ConnectManger 클래스 따로 만들어서 커넥션만 담당하게 했음.

MysqlMain 최종 버전 보면 insertData를 오버로딩해서 매개변수를 어레이리스트로 변경하고 향상된 포문으로 바꿨음 꼭 비교해서 이해 해볼 것!

```
package mc.ys.lesson8;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.ArrayList;

public class MysqlMain {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MysqlMain mm = new MysqlMain();
        mm.test1();
        // try {
        //     mm.test();
        // } catch (ClassNotFoundException e) {
        //     // TODO Auto-generated catch block
        //     e.printStackTrace();
        // } catch (SQLException e) {
        //     // TODO Auto-generated catch block
        //     e.printStackTrace();
        // }
        System.out.println("정상 종료합니다.");
    }

    public void test1() {
        //readyData를 호출하여 100개의 데이터를 가진 list를 가져오는 코드 작성
        ArrayList<GisaDataVO> list = null;
        try {
            list = this.readyData();
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        this.insertData(list.get(0)); // 여기서 this는 sql자신이 객체가 된거니까 this로 호출가능
    }

    private ArrayList<GisaDataVO> readyData() throws FileNotFoundException, IOException {
        // TODO Auto-generated method stub
        ArrayList<GisaDataVO> list = new ArrayList<GisaDataVO>();
        // 파일의 1000라인의 정보를 GisaDataVO를 이용하여 인스턴스로 만들고 ArrayList에 저장
        // 파일에 접속해서 스트림에 연결
    }
}
```

```

File file = new File("./data/Abc1115.txt");
FileReader fr = new FileReader(file);
// 한줄씩 읽기
BufferedReader br = new BufferedReader(fr);
String line = null;
GisaDataVO vo = null;
while((line=br.readLine())!=null) {
    //한줄을 분석해서
    int stdNo = Integer.parseInt(line.substring(0,6));
    String email = line.substring(6,10);
    int kor = Integer.parseInt(line.substring(10,13).trim());
    int eng = Integer.parseInt(line.substring(13,16).trim());
    int math = Integer.parseInt(line.substring(16,19).trim());
    int sci = Integer.parseInt(line.substring(19,22).trim());
    int hist = Integer.parseInt(line.substring(22,25).trim());
    int total = Integer.parseInt(line.substring(25,28).trim());
    String mgrCode = line.substring(28,29);
    String accCode = line.substring(29,30);
    String locCode = line.substring(30,31);
    // vo객체에 할당하고
    vo = new GisaDataVO();
    vo.setStdNo(stdNo);
    vo.setEmail(email);
    vo.setKor(kor);
    vo.setEng(eng);
    vo.setMath(math);
    vo.setSci(sci);
    vo.setHist(hist);
    vo.setTotal(total);
    vo.setMgrCode(mgrCode);
    vo.setAccCode(accCode);
    vo.setLocCode(locCode);
    //리스트에 저장(리스트는 이미 존재해야 함)
    list.add(vo);
}
br.close();
fr.close();
return list;
}

public void insertData(GisaDataVO vo) {
//    //쿼리를 어떻게 구성할까?
//    String sql = "insert into gisatbl values('"+vo.getStdNo()+"')";
//    //위에서 values안에 한줄에 다표시하기엔 너무 많다. 11개나되는데
//    //그렇다고 아래처럼 하기에는 가바지가 계속생산되기에 좋지않음.비효율
//    sql = sql + vo.getEmail();
//    //그래서 방법은? 스트링버퍼를 사용하는 것!! 혹은 스트링빌더
//    //버퍼와 빌더의 차이는 동기화여부. 버퍼가 동기화되어있어서 멀티쓰레드에 안전 -> 데이터보호
//    //하나의 메모리 안에서 내가 필요한 정보를 계속 수정가능
    StringBuffer sql = new StringBuffer("insert into gisatbl values(");
    sql.append(vo.getStdNo()+",");
    sql.append(vo.getEmail()+",");
    sql.append(vo.getKor()+",");
    sql.append(vo.getEng()+",");
    sql.append(vo.getMath()+",");
    sql.append(vo.getSci()+",");
    sql.append(vo.getHist()+",");
    sql.append(vo.getTotal()+",");
    sql.append(vo.getMgrCode()+",");
    sql.append(vo.getAccCode()+",");
    sql.append(vo.getLocCode());
    sql.append(")");
    System.out.println(sql.toString());
}
// 이렇게 스트링버퍼 써도 복잡하다!!

public void test() throws ClassNotFoundException, SQLException {
//서버 접속에 필요한 정보
String url = "jdbc:mysql://localhost:3306/shopdb";
//접속한 뒤에 작업을 하기 위해 필요한 라이브러리 클래스
String driver = "com.mysql.jdbc.Driver"; //클래스임
//아이디 및 비번
String id = "root";
String pwd = "mcys1309";
//드라이버 매니저라는 곳에서 위의 url에접속해서 아이디 비번 가지고 접속
//프로토콜 간의 약속이 있어야 데이터 처리가능한다
//mysql이 만든 드라이버가 보증해주는거. 약속OK - mysql파일받아서 활동OK해줌
Class.forName(driver);
Connection con = DriverManager.getConnection(url,id,pwd);
//커넥션을받아와서 이제 CRUD작업을 할것임 -> 데이터불필요
if(con!=null) {
    System.out.println("connected");
    con.close();
} else {
    System.out.println("fail connection");
}
}

```

```
}
}
```

공유받은 설명! 참고

JDBC 정리

insert에서 데이터가 만약 1000개라치면 커넥션을 1000번 연결했다 끊어야하는 비효율이 있음.

insertData 메소드를 매개변수로 VO가아닌 LIST로 받게끔 오버로딩해서 insert 수정하고 한번에 1000개 입력 가능하게끔 바꿨음

이제 정처기데이터 다 입력했으니 GisaTest파일가서 문제좀 다듬고 1번풀이할거

지금현재 기사테스트 파일은 문제 푸는거 따로 각각 저장하는거는 다른 메소드에서 따로 했는데 해당 문제에서 모두 처리하는 방식으로 수정할 것임

기사문제 1번이랑 나머지문제 위에 말한 것 처럼 리팩토링!

```
package mc.ys.lesson7;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Collections;

public class GisaTest2 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        GisaTest2 gt = new GisaTest2();
        //gt.testString();
        gt.testStart();
    }

    public void testStart() {
        //데이터준비
        try {
            //file의 데이터를 ArrayList로 만들어서 준비
            this.solveOne();
            // this.solveTwo();
            // this.solveThree();
            // this.solveFour();
        } catch (FileNotFoundException ie) {
            System.out.println("정답작성중 FNF에러");
        } catch (IOException e) {
            System.out.println("정답작성중 IO에러");
        }
    }

    private void solveOne() throws FileNotFoundException, IOException {
        // TODO Auto-generated method stub
        ArrayList<GisaDataVO> list = this.readData();
        ArrayList<GisaDataVO> temp = new ArrayList<GisaDataVO>();
        //지역코드 B인 것만 따로 리스트로 구성
        for(GisaDataVO vo : list) {
            if(vo.getLocCode().equals("B")) {
                temp.add(vo); //지역코드가 B인 리스트가 temp
            }
        }
        //정렬하고 싶은 컬렉션 temp, 비교기준 입력. 새로클래스를 만들었음
        //Comparator라는 인터페이스는 compare라는 메소드만 가지고 있어서
        //compare메소드를 구현해야함. 내부에서 정렬하도록.
        Collections.sort(temp, new MyComparator());
        int answer = temp.get(4).getStdNo();
        //인덱스가4니까 5번째출력.
        this.writeAnswer(String.valueOf(answer), 1);
        //this.print(temp);
    }

    private void print(ArrayList<GisaDataVO> temp) {
        // TODO Auto-generated method stub
        for(int i=0;i<10;i++) {
            GisaDataVO vo = temp.get(i);
        }
    }
}
```



```

        System.out.println(vo);
    }
}

private void solveThree() throws FileNotFoundException, IOException {
    // TODO Auto-generated method stub
    ArrayList<GisaDataVO> list = this.readyData();
    String answer = null;
    int sum = 0; //총점+point 누계
    for(GisaDataVO vo : list) {
        int point = 20; //답임코드 C의 값으로 초기화
        if(vo.getEng()+vo.getMath()>=120) {
            if(vo.getMgrCode().equals("A")) {
                point = 5;
            } else if(vo.getMgrCode().equals("B")) {
                point = 15;
            }
            int total = vo.getTotal()+point; //총점+point
            sum = sum +total;
        }
    }
    answer = String.valueOf(sum);
    this.writeAnswer(answer, 3);
}

private void solveFour() throws FileNotFoundException, IOException {
    // TODO Auto-generated method stub
    ArrayList<GisaDataVO> list = this.readyData();
    String answer = null;
    int count = 0; //누적을 저장하는 변수
    for(GisaDataVO vo : list) {
        int point = 15; //지역코드 C의 값으로 초기화
        if(vo.getAccCode().equals("A")||vo.getAccCode().equals("B")) {
            if(vo.getLocCode().equals("A")) {
                point = 5;
            } else if(vo.getLocCode().equals("B")) {
                point = 10;
            }
        }
        int total = vo.getKor()+point;
        if(total>=50) {
            count++;//count += 1, count = count + 1
        }
    }
    answer = String.valueOf(count);
    this.writeAnswer(answer, 4);
}

private void writeAnswer(String answer,int order) throws IOException { //1~4번까지 모두 사용
    // TODO Auto-generated method stub
    // 해당하는 정답을 파일에 작성한다.
    String filePath = "./data/Ans"+order+".txt";
    File file = new File(filePath);
    FileWriter fw = new FileWriter(file);
    PrintWriter pw = new PrintWriter(fw);
    pw.println(answer);
    pw.close();
    fw.close();
}

private void solveTwo() throws IOException {
    // TODO Auto-generated method stub
    ArrayList<GisaDataVO> list = this.readyData();
    String answer = null;
    // 해당하는 문제를 해결하는 로직 작성(최대값 로직)
    int max = 0;
    GisaDataVO vo = null;
    for(int i=0;i<list.size();i++) {
        vo = list.get(i);
        if(vo.getLocCode().equals("B")) {
            if(max<(vo.getKor()+vo.getEng())) {
                max = vo.getKor()+vo.getEng();
            }
        }
    }
    answer = String.valueOf(max);
    this.writeAnswer(answer, 2);
}

private void solveTwoV2() throws FileNotFoundException, IOException {
    // TODO Auto-generated method stub
    ArrayList<GisaDataVO> list = this.readyData();
    String answer = null;
    // 해당하는 문제를 해결하는 로직 작성(최대값 로직)
    int max = 0;
    for(GisaDataVO vo : list) {

```

```

        if(vo.getLocCode().equals("B")) {
            int temp = vo.getKor()+vo.getEng();
            max = max<temp?temp:max;
        }
    }
    answer = String.valueOf(max);
    this.writeAnswer(answer, 2);
}

private ArrayList<GisaDataVO> readyData() throws FileNotFoundException, IOException {
    // TODO Auto-generated method stub
    ArrayList<GisaDataVO> list = new ArrayList<GisaDataVO>();
    // 파일의 1000라인의 정보를 GisaDataVO를 이용하여 인스턴스로 만들고 ArrayList에 저장
    // 파일에 접속해서 스트림에 연결
    File file = new File("./data/Abc1115.txt");
    FileReader fr = new FileReader(file);
    // 한줄씩 읽기
    BufferedReader br = new BufferedReader(fr);
    String line = null;
    GisaDataVO vo = null;
    while((line=br.readLine())!=null) {
        //한줄을 분석해서
        int stdNo = Integer.parseInt(line.substring(0,6));
        String email = line.substring(6,10);
        int kor = Integer.parseInt(line.substring(10,13).trim());
        int eng = Integer.parseInt(line.substring(13,16).trim());
        int math = Integer.parseInt(line.substring(16,19).trim());
        int sci = Integer.parseInt(line.substring(19,22).trim());
        int hist = Integer.parseInt(line.substring(22,25).trim());
        int total = Integer.parseInt(line.substring(25,28).trim());
        String mgrCode = line.substring(28,29);
        String accCode = line.substring(29,30);
        String locCode = line.substring(30,31);
        // VO객체에 할당하고
        vo = new GisaDataVO();
        vo.setStdNo(stdNo);
        vo.setEmail(email);
        vo.setKor(kor);
        vo.setEng(eng);
        vo.setMath(math);
        vo.setSci(sci);
        vo.setHist(hist);
        vo.setTotal(total);
        vo.setMgrCode(mgrCode);
        vo.setAccCode(accCode);
        vo.setLocCode(locCode);
        //리스트에 저장(리스트는 이미 존재해야 함)
        list.add(vo);
    }
    br.close();
    fr.close();
    return list;
}

public void testString() {
    String line = "990001addx 17 29 16 49 43154CAC";
    //0123456789012345678901234567890
    //6 4 3 3 3 3 3 1 1 1
    //학번과 국어점수를 분리하여 지시문에서 선언된 타입으로 변수에 할당하시오.
    int stdNo = Integer.parseInt(line.substring(0,6));
    int kor = Integer.parseInt(line.substring(10,13).trim());
    System.out.printf("stdNo = %d kor = %d %n", stdNo,kor);
}
}

```

오늘한 것 정리

테이블작성쿼리(ddl) pk,fk,default, auto_increment, dml(insert,delete)

어플리케이션 DB접속 JDBC - dml

jdbc에 대해 제대로 이해하고 외울거 외우고 가야함!! 중요

인터페이스를 이용한 정처기문제1번 해결. + 클래스 구조를 수정변경하는 작업.

내일은 select를 해볼거임 jdbc로. 제일 중요하다

[TIL] JDBC로 DB(MySQL) 연결해서 사용하기 - SELECT, INSERT 예제 / java.sql.SQLException: Column count doesn't match value count at row 숫자.
오늘 저녁에는 어제에 이어서 JDBC로 MySQL DB 연결을 연습해 보았다! (SELECT 와 INSERT) * 링크1, 링크2를 참조하며 연습한 예제입니다. 감사합니다 🙏 특히 링크1 중심으로 따라한 것이 많은 도움이 되었습니다.

👤 <https://bibi6666667.tistory.com/193>

예제코드가 너무 길어서 링크 참고해서 보면 될듯. 아메가메님이 정리해주신 파일이랑 유튜브 보니까 이해는 됐다! 안보고 작성을 못할 뿐...

@2022년 1월 20일

#1/20

select 쿼리문 전달하는 거 해봄

ResultSet은 래퍼클래스를 상속받아 테이블의 데이터 타입이 무엇이든 간에 스트링으로 호출 가능.

기사문제 쿼리문으로 풀어봤음!

```
-- 기사문제 쿼리로 풀기
-- 지역코드가 B인 레코드중 국어+영어 내림차, 동점이면 학번으로 오름차순
select std_no, (kor+eng)
from gisatbl
where loc_code = 'B' order by (kor+eng) desc,
std_no asc limit 4,1; -- 인덱스가 4부터해서 1개만 보여주세요
-- limit 5하면 5개출력하는데 4,1하면 인덱스 4부터 1개
-- limit 3,3 하면 4번째 자료부터 3개 출력
-- 그냥 limit 5 숫자는 개수인데 두개쓰면 인덱스가됨

select max(kor+eng) from gisatbl where loc_code = 'B';

select sum(total+
case
    when mgr_code = 'A' then 5
    when mgr_code = 'B' then 15
    when mgr_code = 'C' then 20
end
) as '자료의 합'
from gisatbl where eng+math >= 120;

select sum((kor+
case
    when loc_code = 'A' then 5
    when loc_code = 'B' then 10
    when loc_code = 'C' then 15
end) >= 50 ) V
from gisatbl where (acc_code = 'A' or acc_code='B');

-- 위 문제랑 같은 문제 다른 방법
select count(*)
from (
select (kor+
case
    when loc_code = 'A' then 5
    when loc_code = 'B' then 10
    when loc_code = 'C' then 15
end) B
from gisatbl where acc_code = 'A' or acc_code = 'B') V
where V.B >= 50;
-- from ( ) 서브쿼리를 사용한건데 셀렉트 구문은 내부적으로 테이블 형태
-- 테이블 형태이니까 v라고 이름 지어줄 수 있다.
-- 테이블 형태여서 from뒤나 where자리에 서브쿼리로 쓸수있다.
-- case랑 end는 세트이고 책에 case뒤에 값이 있는데
-- when안에 들어가있는 형태라고 보면 되네
-- 여기서 when은 if라고 생각하면되고 여러개니까
-- if, else if, else if 형태인거. 어느 조건에도 들어가지 않으면 else추가하면됨
```

Prepared Statement

- 테이블에 객체 저장할 때 필요! 문자열데이터로 바뀌서 테이블로 저장했는데 객체는 저장할 방법이 없다!
- 입력할 데이터만 변경이 될 때
- 데이터의 무결성 확인 - 데이터 타입을 확실하게 확인해서 보냄. 원래 쿼리 보낼때 "100" + "100"을 날리면 원래 안되어야하는데 int로 알아서 잘들어갔음. 무결성이랑 조금 어긋나는 부분. 그래서 prepared statement써서 체크해주는 것. 애사용하면 위에처럼 안되고 칼같이 함.

그냥 statement쓰면 "연월일 시분초"이렇게 해도 datetime타입으로 자동으로 들어간다는 말임.

무결성을 위해 타입을 맞춰줄것임!

우리가 처음 JDBC 인서트할 때

1. 하드코딩. 코딩의 목표는 하드코딩을 최소화하는 것. 일일이 데이터를 ""사용해서 입력하기 어렵고 오류 발생확률이 높다.
2. ArrayList, StringBuffer 를 사용하면 따옴표도 안써도되고 오류도 줄이면서 쉽게 입력할 수 있음. 하지만 객체를 넣을 수 없고, 데이터 타입도 무조건 String으로 넣어야해서 무결성에도 맞지 않음. 컴파일도 매번 계속해야함. 비효율적
3. 그래서 마지막 방법인 PreparedStatement를 쓰는것!

```
public void testInsertV2() throws SQLException {
    String sql = "insert into emp(emp_name, emp_age,emp_hiredate,dept_code) values(?,?,?,?)";
    // ?는 플레이스 홀더. 4개의 데이터가 올건데 아직 몰라
    ConnectionManager cm = new ConnectionManager();
    Connection con = cm.getConnection();
    //통로. sql이 준비됐다. 데이터를 제외한 나머지는 컴파일 된상태고
    //? ? ? ? 나머지 값만 넣음.
    PreparedStatement pstmt = con.prepareStatement(sql);
    pstmt.setString(1, "Lee");
    pstmt.setInt(2, 20);
    Timestamp stamp = new Timestamp(System.currentTimeMillis());
    pstmt.setTimestamp(3, stamp);
    //pstmt.setString(3,"2022-01-20 14:01:20");
    //시간타입인데 String타입으로 X. 게다가 시간도 하드코딩!
    //Timestamp라는 객체를 넣어준 것!
    pstmt.setString(4, "ABCDE"); //varchar도 문자열이니 String으로
    int affectedCount = pstmt.executeUpdate();
    // 이미 pstmt에서 쿼리를 넣어 보냈으니까 여기서 업데이트 안에 sql 안넣음!!
    if(affectedCount>0) {
        System.out.println("삽입완료");
    } else {
        System.out.println("삽입실패");
    }
    cm.closeConnection(con, pstmt, null); //리자셋이없으니 null
}
```

pstmt는 prepare할때 데이터 파트 ?부분만 남겨두고 컴파일함. 그래서 executeUpdate()는 무인자.

데이터타입도 호환이 되는 데이터타입만 넣을 수 있는 통로이기 때문에 timestamp 객체를 생성해서 넣어준거. 객체를 저장할 수 있다.

statement는 데이터가 계속 변경이 되므로 컴파일을 매번해야하는데 preparedStatement는 입력할 데이터만 ?에서 변경이 된다! 컴파일 1번!

Statement는 프리페어드스태이트먼트의 조상. 하는일이 스테이트먼트가 더 하는 일이 적음.

근데 만약 몇줄 그냥 입력하고 타입도 별로 고려안하고 string으로 보내면되는 쿼리를 전달할 때에는 굳이 메모리 크기를 많이 차지하는 프리페어드를 사용하는 것보다 statement사용이 효율적일 때 사용한다 정도로 생각하고 있으면 된다.

정리하자면 pstmt사용이유

1. 테이블에 객체를 저장하기 위해. statement는 무인자. 객체를 저장할 수 없다. pstmt는 연결하면서 정해지지않은 데이터 ?부분만 남기고 sql을 전달함.
2. 입력할 데이터만 변경해서 컴파일 1번에 끝내고 효율성을 높이기 위해
3. 데이터의 무결성을 지키기 위해.

executeQuery(): select문 실행. executeUpdate(): insert/update/delete문 실행. rs.next(): select 문 결과 행에서 커서를 이동하면서 다음 행 지정. rs.getString()/rs.getInt(): 다음 행(레코드)이 존재하면 데이터 가져오기.

DB기간 끝!

직접 DTO, DAO, 실행 클래스 만들어서 DB연동하는 과정 거치니 어느정도 JDBC의 정형화된 코드에 익숙해졌다. 까먹지 않게 자주 복습 할 것!!