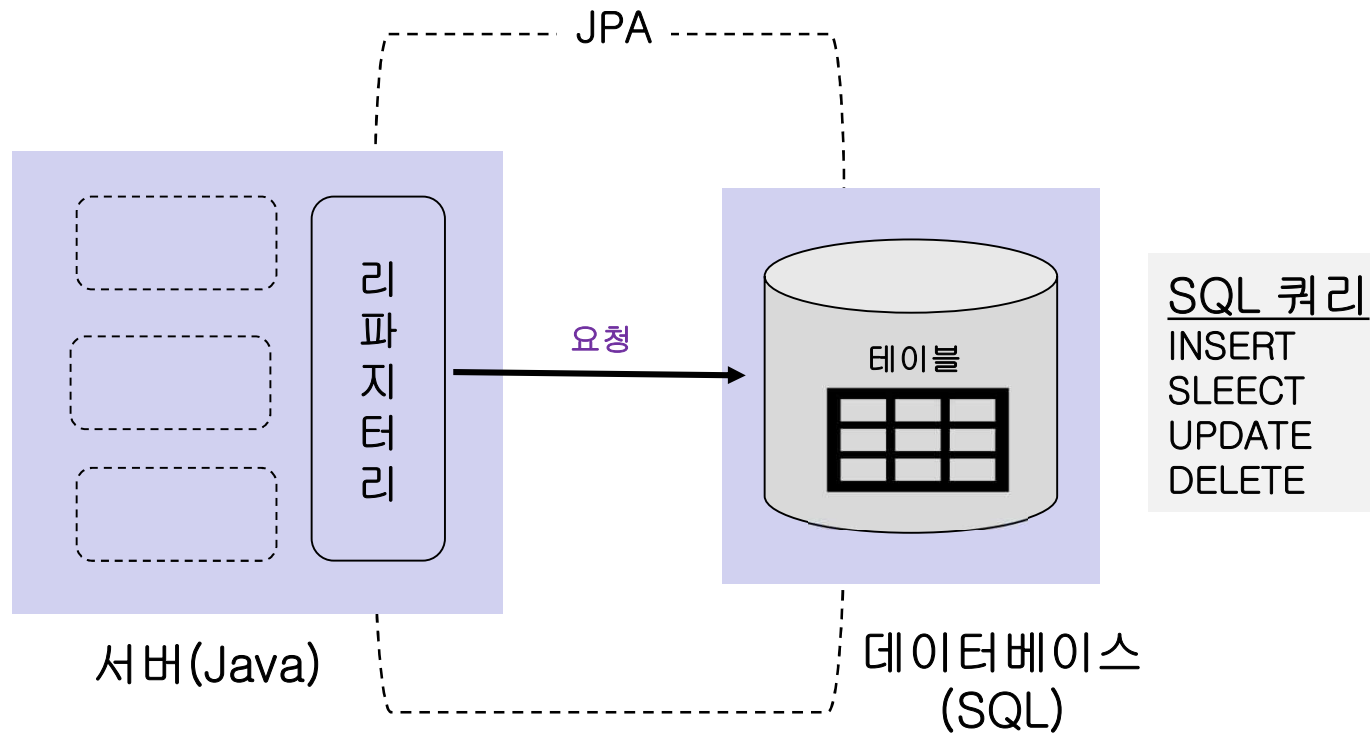


# 9장 CRUD와 SQL 쿼리 종합

출처: 코딩 자율학습 스프링부트3 자바 백엔드 개발 입문, 홍팩, 길벗, 2023

# CRUD 수행에 따른 SQL 쿼리



# JPA 로깅 설정하기

- 로깅 레벨

- TRACE(레벨1): DEBUG 레벨보다 더 상세한 정보
- DEBUG(레벨2): 응용 프로그램을 디버깅하는 데 필요한 세부 정보
- INFO(레벨3): 응용 프로그램의 순조로운 진행 정보
- WARN(레벨4): 잠재적으로 유해한 상황 정보
- ERROR(레벨5): 응용 프로그램이 수행할 수 있는 정도의 오류 정보
- FATAL(레벨6): 응용 프로그램이 중단될 만한 심각한 오류 정보
- OFF(레벨7): 로깅 기능 해제

# JPA 로깅 설정하기

- application.properties

```
spring.application.name=firstproject
server.servlet.encoding.force=true
spring.h2.console.enabled=true
spring.jpa.defer-datasource-initialization=true

# JPA 로깅 설정
# 디버그 레벨 쿼리 출력
logging.level.org.hibernate.SQL=DEBUG
```

# JPA 로깅 설정하기

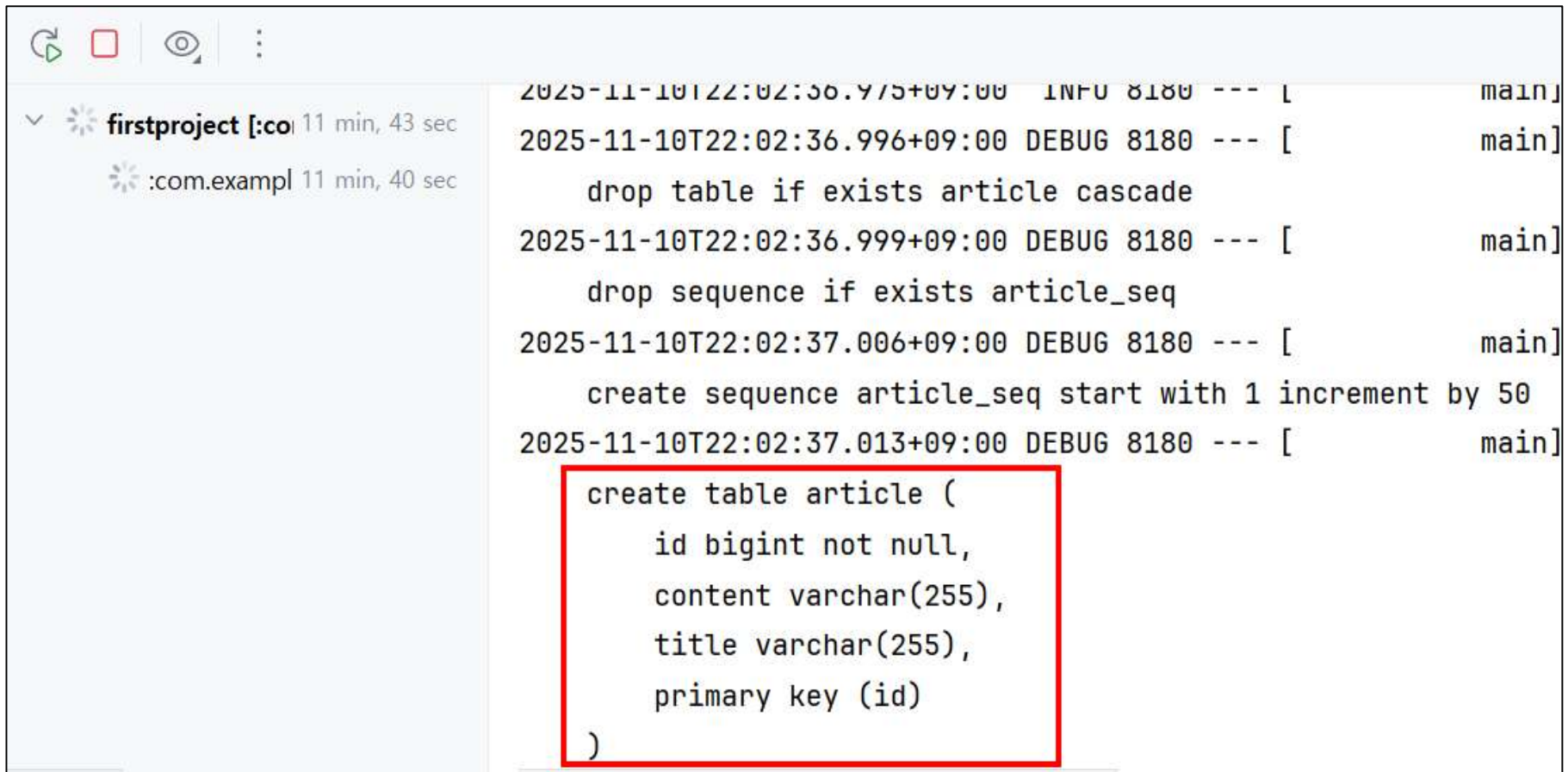
- SQL 쿼리 로그에 줄바꿈 적용
  - application.properties

```
spring.application.name=firstproject
server.servlet.encoding.force=true
spring.h2.console.enabled=true
spring.jpa.defer-datasource-initialization=true

# JPA 로깅 설정
# 디버그 레벨 쿼리 출력
logging.level.org.hibernate.SQL=DEBUG
# 쿼리 줄바꿈하기
spring.jpa.properties.hibernate.format_sql=true
```

# JPA 로깅 설정하기

- 서버 재실행, 로그 확인



```
2025-11-10T22:02:36.975+09:00 INFO 8180 --- [main]
2025-11-10T22:02:36.996+09:00 DEBUG 8180 --- [main]
    drop table if exists article cascade
2025-11-10T22:02:36.999+09:00 DEBUG 8180 --- [main]
    drop sequence if exists article_seq
2025-11-10T22:02:37.006+09:00 DEBUG 8180 --- [main]
    create sequence article_seq start with 1 increment by 50
2025-11-10T22:02:37.013+09:00 DEBUG 8180 --- [main]
    create table article (
        id bigint not null,
        content varchar(255),
        title varchar(255),
        primary key (id)
    )
```

# JPA 로깅 설정하기

- JPA 쿼리에서 DB로 넘어가는 매개변수 값을 로그에서 확인
  - Sql 쿼리의 ?(물음표)에 들어 있는 값을 볼 수 있도록 설정
  - application.properties
    - ✓ 구버전(hibernate 5 일 때)
      - logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE
    - ✓ spring boot 3이상, hibernate 6
      - logging.level.org.hibernate.orm.jdbc.bind=TRACE

```
# JPA 로깅 설정
# 디버그 레벨 쿼리 출력
logging.level.org.hibernate.SQL=DEBUG
# 쿼리 줄바꿈하기
spring.jpa.properties.hibernate.format_sql=true
# 매개변수 값 보여 주기
logging.level.org.hibernate.orm.jdbc.bind=TRACE
```

# H2 DB, 고정 URL 사용

- application.properties

(중략)

# 매개변수 값 보여 주기

logging.level.org.hibernate.orm.jdbc.bind=TRACE

# DB URL 설정

# 유니크 URL 생성하지 않기

spring.datasource.generate-unique-name=false

# 고정 URL 설정하기

spring.datasource.url=jdbc:h2:mem:testdb

# SQL 쿼리 로그 확인하기

- 데이터 생성 시: insert 문
- 데이터 조회 시: select 문
- 데이터 수정 시: update 문
- 데이터 삭제 시: delete 문

# 데이터 생성 시: insert 문

- localhost:8080/articles/new – 게시물 작성 – submit  
→ 에러 페이지 (server error: 500)

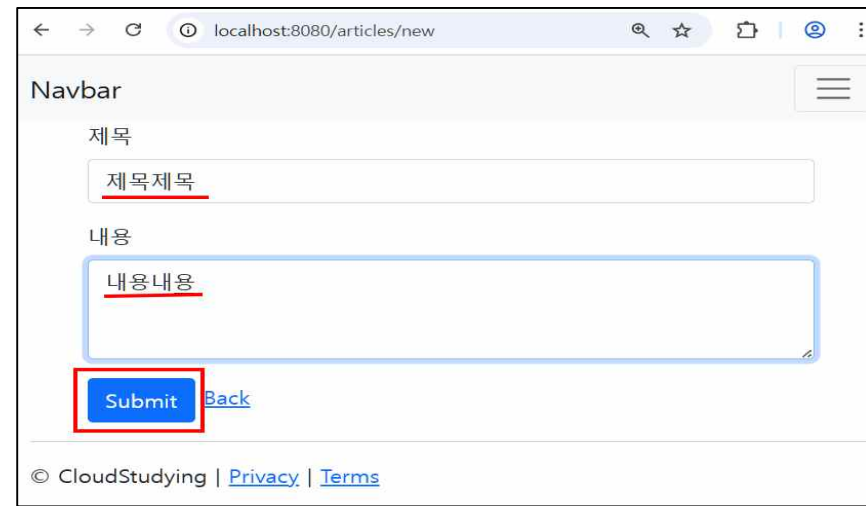


Navbar

Id	Title	Content
1	<a href="#">가가가가</a>	1111
2	<a href="#">나나나나</a>	2222
3	<a href="#">다다다다</a>	3333

[New Article](#)

© CloudStudying | [Privacy](#) | [Terms](#)



Navbar

제목

내용

[Back](#)

© CloudStudying | [Privacy](#) | [Terms](#)



## Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Tue Nov 11 17:51:25 KST 2025  
There was an unexpected error (type=Internal Server Error, status=500).

**Unique index or primary key violation:** "PRIMARY KEY ON PUBLIC.ARTICLE(ID) ( /\* key:1 \*/ CAST(1 AS BIGINT), '1111', U&'WWac00WWac00WWac00WWac00')"; SQL statement:  
  
insert into article (content,title,id) values (?, ?, ?) [23505-214]

# 데이터 생성 시: insert 문

- Id 자동 생성 전략 추가하기

- DB가 알아서 AUTO\_INCREMENT 기능을 사용하여 기본 키를 생성
- entity/Article.java

```
(중략)
@NoArgsConstructor
@AllArgsConstructor
@ToString
@Getter
@Entity
public class Article {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column
    private String title;
    @Column
    private String content;
}
```

# 데이터 생성 시: insert 문

- Id 자동 생성 전략 추가하기(Cont')

- data.sql에서 id 속성과 그 값을 삭제

```
INSERT INTO article(title, content) VALUES ('가가가가', '1111');  
INSERT INTO article(title, content) VALUES ('나나나나', '2222');  
INSERT INTO article(title, content) VALUES ('다다다다', '3333');
```

- 서버 재실행

- localhost:8080/articles/new
  - ✓ 제목, 내용 입력 후, submit
  - ✓ 실행창에서 sql 로그 확인

```
2025-11-11T18:31:48.253+09:00 DEBUG 20816 --- [nio-8080-exec-4] org.hibernate.SQL :  
insert  
into  
  article  
  (content,title,id)  
values  
  (?,?,default)  
(중략): binding parameter [1] as [VARCHAR] - [내용내용]  
(중략) : binding parameter [2] as [VARCHAR] - [제목제목]
```

# 데이터 조회 시: select 문

- **localhost:8080/articles**

- sql 로그 확인

```
select
  a1_0.id,
  a1_0.content,
  a1_0.title
from
  article a1_0
```

- **localhost:8080/articles/1**

- Sql 로그 확인

```
select
  a1_0.id,
  a1_0.content,
  a1_0.title
from
  article a1_0
where
  a1_0.id=?
(중략) : binding parameter [1] as [BIGINT] - [1]
```

# 데이터 수정 시: update 문

- localhost:8080/articles/1/edit
  - Sql 로그 확인

```
update
  article
set
  content=?,
  title=?
where
  id=?
```

(중략): binding parameter [1] as [VARCHAR] - [1111abc]

(중략): binding parameter [2] as [VARCHAR] - [가가가가abc]

(중략): binding parameter [3] as [BIGINT] - [1]

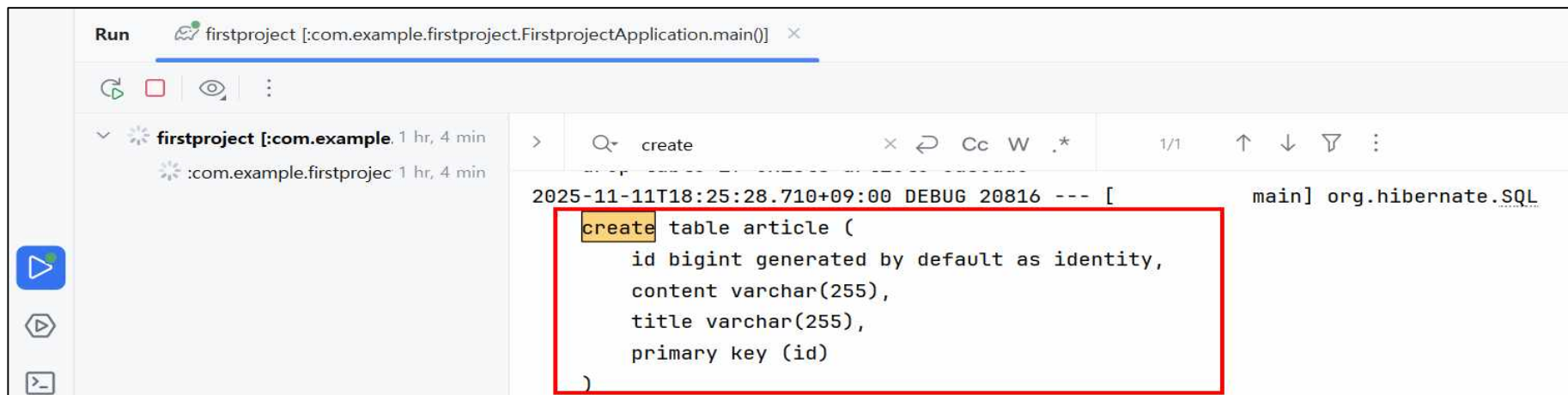
# 데이터 삭제 시: delete 문

- localhost:8080/articles/1 접속 후, [Delete] 클릭
  - Sql 로그 확인

```
delete
  from
    article
  where
    id=?
```

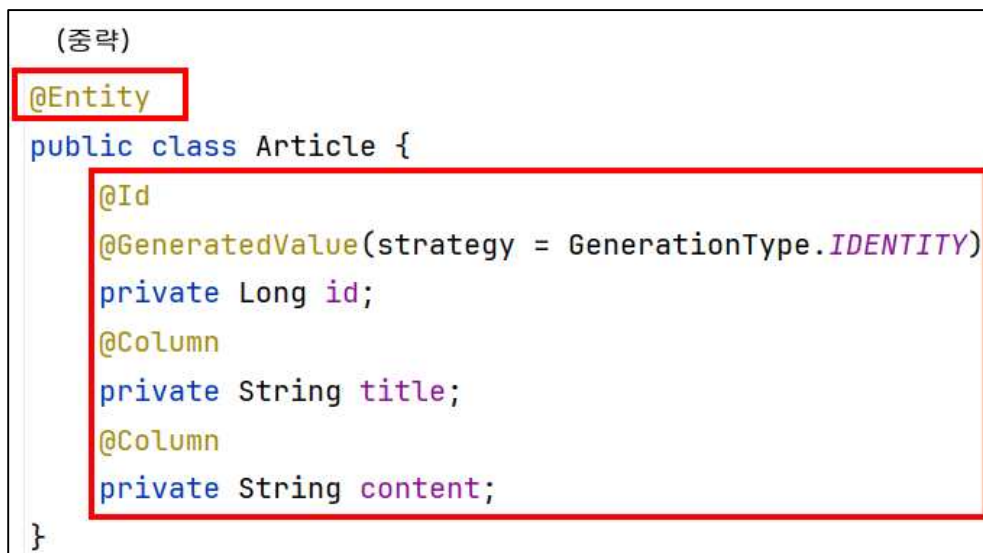
(중략) : binding parameter [1] as [BIGINT] - [1]

- SQL 쿼리 로그에서 CREATE 검색



```
Run firstproject [com.example.firstproject.FirstprojectApplication.main()] x
firstproject [com.example.firstproject.FirstprojectApplication.main()] 1 hr, 4 min
:com.example.firstproject.FirstprojectApplication.main() 1 hr, 4 min
> create
2025-11-11T18:25:28.710+09:00 DEBUG 20816 --- [main] org.hibernate.SQL
create table article (
  id bigint generated by default as identity,
  content varchar(255),
  title varchar(255),
  primary key (id)
)
```

- entity/Article.java



```
(중략)
@Entity
public class Article {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column
    private String title;
    @Column
    private String content;
}
```

서버를 실행하면  
새로운 Article 객체가  
만들어지기 때문에  
CREATE TABLE 문도 자동 수행됨

# 기본 SQL 쿼리 작성하기

- localhost:8080/h2-console 접속

English ▾ Preferences Tools Help

Login

Saved Settings: Generic H2 (Embedded) ▾

Setting Name: Generic H2 (Embedded) Save Remove

Driver Class: org.h2.Driver

JDBC URL: jdbc:h2:mem:testdb

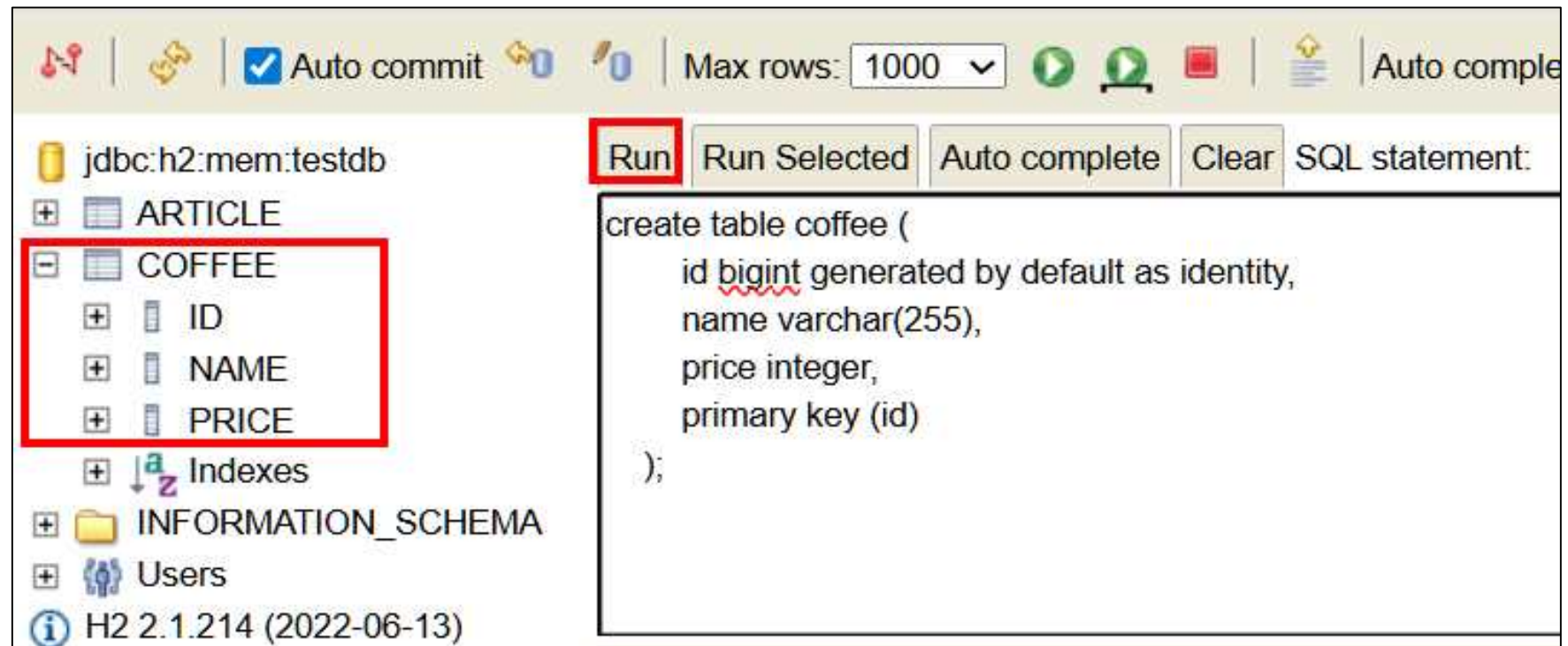
User Name: sa

Password:

Connect Test Connection

# 기본 SQL 쿼리 작성하기

- coffee 테이블 생성



# 기본 SQL 쿼리 작성하기

- **SELECT \* FROM COFFEE;**
- **INSERT INTO coffee ( id, name, price ) VALUES (1, '아메리카노', 4100);**
- **INSERT INTO coffee ( id, name, price )  
VALUES (2, '라떼', 4600), (3, '모카', 5100), (4, '오늘의 커피', 3800);**
- **SELECT id, name, price FROM coffee WHERE id =3;**
- **UPDATE coffee SET price=9900 WHERE id = 4;**
- **DELETE FROM coffee WHERE id=4;**