



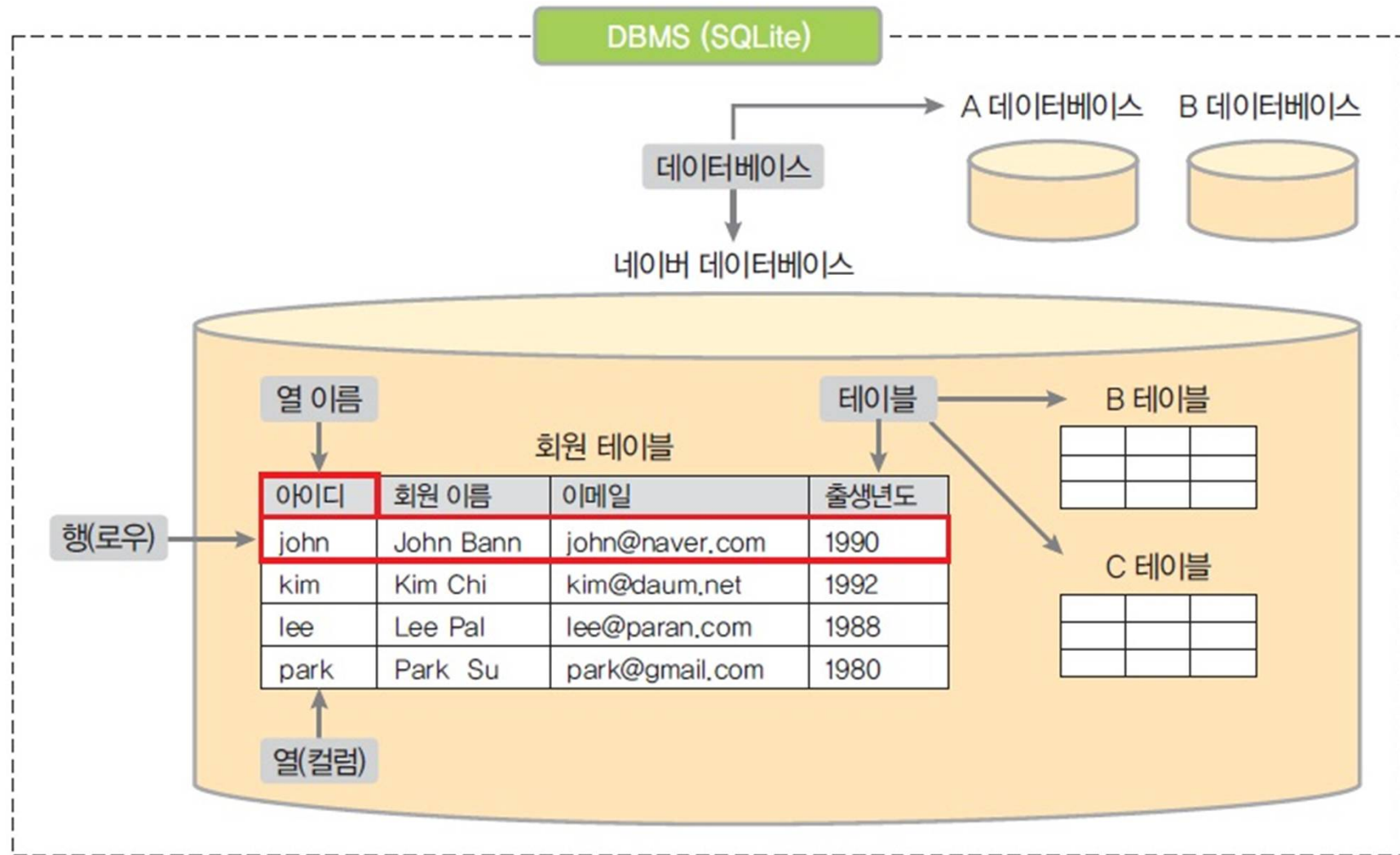
데이터베이스

- 용어 정리
 - DBMS란?
 - 데이터베이스란?
 - 테이블이란?
 - 열이란?
 - 행이란?
- SQL = DML(데이터 조작) + DDL(테이블 조작) + DCL(제어)
- DML : CRUD2SG
 - C: INSERT
 - R: SELECT
 - U: UPDATE
 - D: DELETE
 - S: SELECT ~ WHERE
 - S: SELECT ~ ORDER BY
 - G: GROUP BY ~ HAVING

데이터베이스
를 사용하는
방법을
학습합니다.



데이터베이스 기본 개념



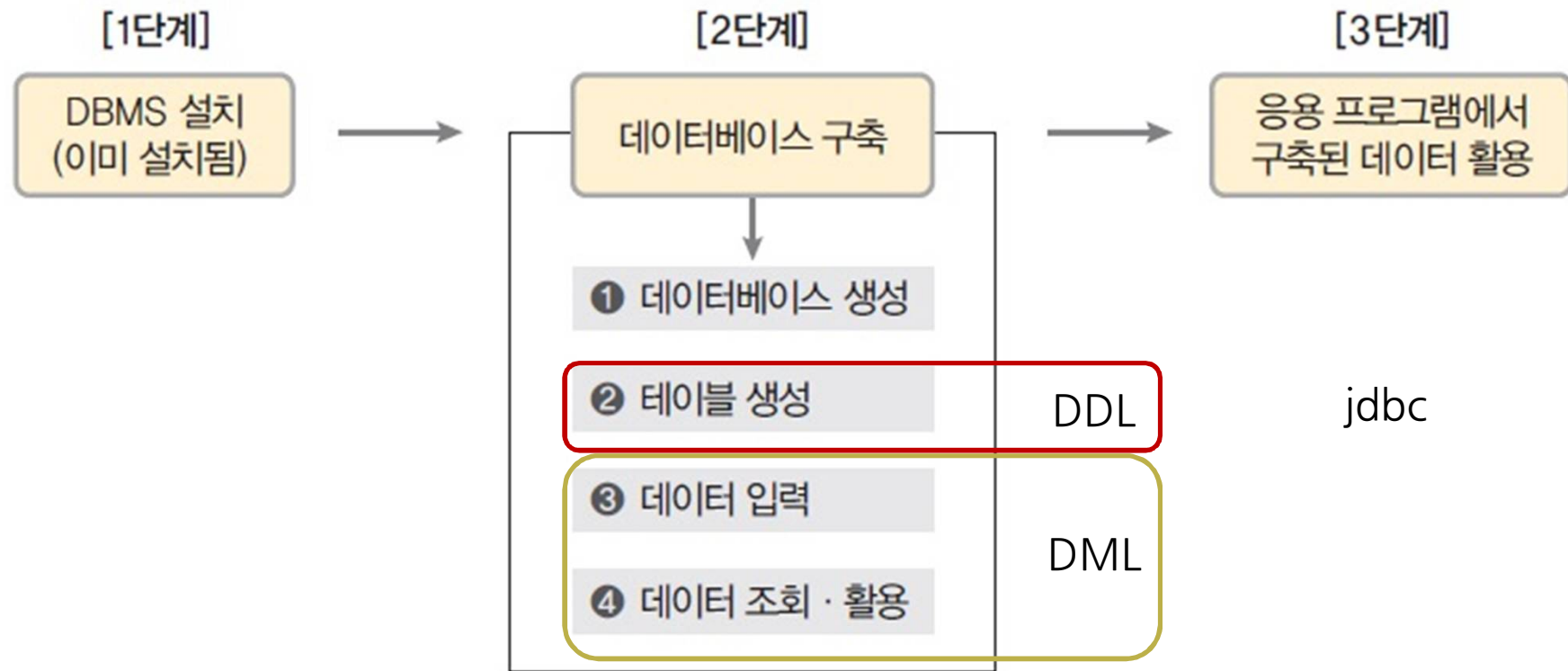


데이터베이스 기본 개념

- 데이터베이스 관련 용어
 - **데이터** : 하나하나의 단편적인 정보를 뜻함
 - **DBMS** : 데이터베이스를 관리하는 소프트웨어를 말함
Oracle, MS-SQL, MySQL, SQLite 소프트웨어가 이에 해당
 - **데이터베이스(DB)** : 테이블이 저장되는 장소로 원통 모양으로 표현
각 데이터베이스는 서로 다른 고유한 이름이 있어야 함
 - **테이블** : 데이터가 표 형태로 표현된 것
 - **열(컬럼 또는 필드)** : 각 테이블은 1개 이상의 열로 구성됨
 - **열 이름** : 각 열을 구분하는 이름, 열 이름은 각 테이블 안에서는 중복되지 않아야 함
 - **데이터 형식** : 열의 데이터 형식을 뜻함
숫자, 문자, 날짜
테이블을 생성할 때 열 이름과 함께 지정해줘야 함
 - **행(로우)** : 실제 데이터를 뜻함
 - **SQL** : 사용자와 DBMS와 소통하기 위해서 사용하는 언어
 $\text{SQL} = \text{DML(CRUD2SG)} + \text{DDL} + \text{DCL}$



데이터베이스 구축





Introduction

- DBMS
 - Database management system
 - Storing and organizing data
- Database
 - Collection of data
- Table
- Column
- Row, Record
- SQL
 - Relational database
 - Structured Query Language
- JDBC
 - Java Database Connectivity
 - JDBC driver



SQL이란?

- SQL = Structured Query Language
- 데이터베이스에서 사용하기 위하여 설계된 언어
- DDL + DML + DCL
 - DDL : Data Definition Language : DB, 테이블 조작
 - DML : Data Manipulation Language : 테이블에서 데이터 조작
 - DCL : Data Control Language : DB 제어



DDL이란?

데이터베이스나 테이블을 생성하거나 수정하는 데 사용되는 언어.

- 생성(CREATE),
- 변경(ALTER),
- 제거(DROP)
- 권한 부여(GRANT)
- 권한 박탈(REVOKE)

구분	명령어	설명
데이터 정의 명령어 (Data Definition Language)	CREATE	사용자가 제공하는 컬럼 이름을 가지고 테이블을 생성한다. 사용자는 컬럼의 데이터 타입도 지정하여야 한다. 데이터 타입은 데이터베이스에 따라 달라진다. CREATE TABLE은 보통 DML보다 적게 사용된다. 왜냐하면 이미 테이블이 만들어져 있는 경우가 많기 때문이다.
	ALTER	테이블에서 컬럼을 추가하거나 삭제한다.
	DROP	테이블의 모든 레코드를 제거하고 테이블의 정의 자체를 데이터베이스로부터 삭제하는 명령어이다.
	USE	어떤 데이터베이스를 사용하는지를 지정



데이터베이스 접속하기

HS 세션 관리자

세션 이름 ▲
localhost

설정 고급 통계

네트워크 유형: MySQL (TCP/IP)

호스트명 / IP: localhost

☐ 자격 증명 프롬프트
☐ Windows 인증 사용

사용자: root

암호: ●●●●

포트: 3306

☐ 압축된 클라이언트/서버 프로토콜

데이터베이스: 세미콜론으로 구분

코멘트:

신규 ▼ 저장 삭제 열기 취소 더 보기 ▼



데이터베이스 생성하기

-- 데이터베이스 삭제

```
DROP DATABASE IF EXISTS book_db;
```

-- 데이터베이스 생성

```
CREATE DATABASE book_db COLLATE 'utf8_general_ci';
```

-- 데이터베이스 사용

```
USE book_db;
```



데이터베이스 생성하기

localhostW - HeidiSQL Portable 9.3.0.4984

파일 편집 검색 도구 도움말

데이터베이스 필터 테이블 필터

localhost

- information_sc
- mysql
- performance_s

호스트: localhost 쿼리

데이터베이스 (5) 변수

1 새로 생성(O) 2 데이터베이스(T)

데이터베이스 생성...

이름(N) book_db

조합(O) utf8_general_ci

서버 기본값: latin1_swedish_ci

확인 취소

CREATE 코드:

```
CREATE DATABASE `book_db` /
```

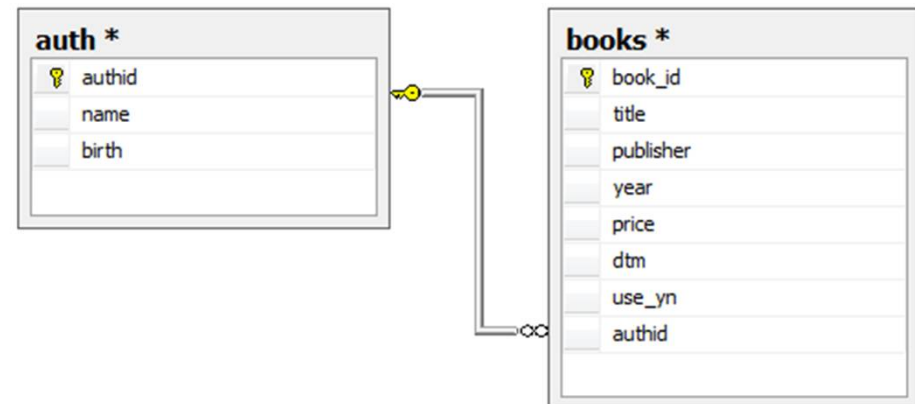


테이블 생성하기

-- 테이블 생성

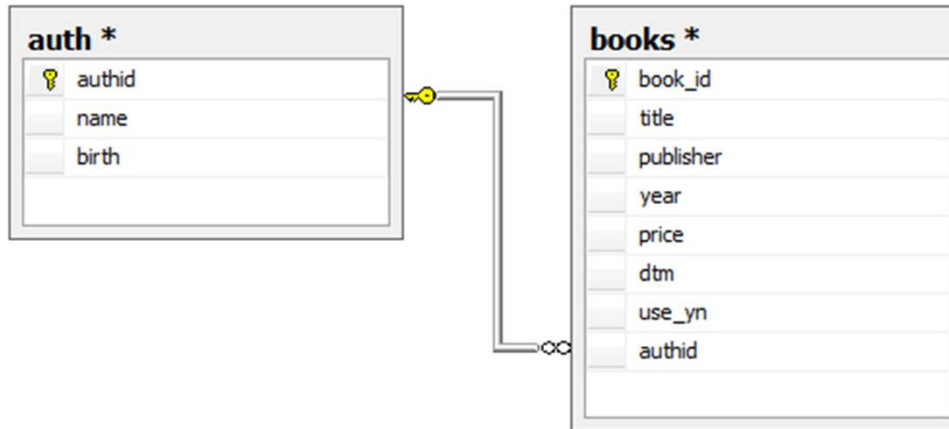
```
DROP TABLE IF EXISTS book;
```

```
CREATE TABLE book (  
    bookid          INT NOT NULL auto_increment  
    , bookname      VARCHAR(50)  
    , publisher     VARCHAR(30)  
    , year          VARCHAR(10)  
    , price         INT  
    , dtm           DATE  
    , use_yn        BIT  
    , authid        INT NOT NULL  
  
    , PRIMARY KEY(bookid)  
)  
COLLATE='utf8_general_ci'  
ENGINE=InnoDB ;
```





테이블 생성하기



-- 테이블 생성

```
DROP TABLE IF EXISTS auth;
```

```
CREATE TABLE auth (
    authid      INT NOT NULL auto_increment
    , name      VARCHAR(50)
    , birth     VARCHAR(10)

    , PRIMARY KEY(authid)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB ;
```



테이블 생성하기

1

2

테이블: [Untitled]

1

2

추가

#	이름	데이터 유형	길이/설정	부호 없음	NULL 허용	0으로 채움	기본값
---	----	--------	-------	-------	---------	--------	-----

Detailed description: The image shows a screenshot of a database management tool interface. On the left, a tree view shows a database named 'book_db'. A right-click context menu is open over 'book_db', with '새로 생성(O)' (New) highlighted by a red box and a yellow '1'. A sub-menu is open for '새로 생성(O)', with '테이블(U)' (Table) highlighted by a red box and a yellow '2'. Below the menu, the '테이블: [Untitled]' tab is active. In the '이름:' (Name) field, '테이블 이름 입력' (Enter table name) is entered, highlighted by a red box and a yellow '1'. Below this, the '열:' (Columns) section has a '+ 추가' (Add) button highlighted by a red box and a yellow '2'. At the bottom, a table structure grid is visible with columns for #, 이름 (Name), 데이터 유형 (Data Type), 길이/설정 (Length/Settings), 부호 없음 (Unsigned), NULL 허용 (Allow Null), 0으로 채움 (Zero-filled), and 기본값 (Default Value).



테이블 Primary Key 생성하기

기본 옵션 인덱스 외래 키 분할 CREATE 코드 ALTER 코드

이름: tb_saved

열: + 추가 - 제거 ▲ 위로 ▼ 아래로

#	이름	데이터 유형	길이/설정	부호 없음	NULL 허용	0으로 채움	기본값
1	saved_id			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT
2	name			<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
3	expression			<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL

Context menu for 'saved_id':

- 복사(C) Ctrl+C
- 선택한 열 복사(S)
- 열 붙여넣기(T)
- 열 추가(U) Ctrl+Ins
- 열 제거(V) Ctrl+Del
- 위로(W) Ctrl+U
- 아래로(X) Ctrl+D
- 새 인덱스 생성(Y)
 - PRIMARY
 - KEY
 - UNIQUE
 - FULLTEXT
 - SPATIAL
- 인덱스에 추가(Z)



DML이란?

데이터를 실질적으로 처리하는 데 사용되는 언어입니다. (SUDI)

- SELECT(검색)
- UPDATE(수정)
- INSERT(삽입)
- DELETE(삭제)
- COMMIT(트랜잭션)
- ROLLBACK(트랜잭션)

구분	명령어	설명
데이터 조작 명령어 (Data Manipulation Language)	SELECT	데이터베이스로부터 데이터를 쿼리하고 출력한다. SELECT 명령어들은 결과 집합에 포함시킬 컬럼을 지정한다. SQL 명령어 중에서 가장 자주 사용된다.
	INSERT	새로운 레코드를 테이블에 추가한다. INSERT는 새롭게 생성된 테이블을 채우거나 새로운 레코드를 이미 존재하는 테이블에 추가할 때 사용된다.
	DELETE	지정된 레코드를 테이블로부터 삭제한다.
	UPDATE	테이블에서 레코드에 존재하는 값을 변경한다.



레코드 검색하기

```
SELECT * FROM book;
```

```
SELECT * FROM book WHERE bookname like '%SQL%';
```

```
SELECT * FROM book WHERE bookname = 'JAVA';
```

```
SELECT * FROM book WHERE 30700 <= price and price <50000;
```

```
SELECT * FROM book WHERE price BETWEEN 30700 AND 50000;
```

```
SELECT * FROM book WHERE price <= 30700 OR 58000 < price ;
```

```
SELECT bookname, publisher, price, authid FROM book;
```




레코드 정렬하기

```
SELECT * FROM book ORDER BY price ASC;
```

```
SELECT * FROM book ORDER BY price DESC;
```

```
SELECT * FROM book ORDER BY publisher ASC, price DESC;
```



레코드 추가하기

```
INSERT 테이블명( 컬럼명1, 컬럼명2, 컬럼명3, .. , 컬럼  
명N )
```

```
VALUES( 값1      ,      값2,      값3, .. , 값
```

```
INSERT INTO book (bookname  , publisher, year  , price, authid )  
VALUES('System', 'Wiley'   , '2003', 30700, 1      ) ;
```

```
INSERT INTO auth (authid, name  , birth      )  
VALUES(1      , 'bob'  , '1970.05.01' ) ;
```



레코드 수정하기

```
/* UPDATE 테이블명  
    SET 컬럼명1=값1  
    , ..  
    , 컬럼명N=값N  
    WHERE 컬럼명=값 ; */
```

```
UPDATE book  
    SET year = '2016'  
    WHERE bookname like '%SQL%' ;
```

```
UPDATE book  
    SET year = '2010'  
    , price = 20000  
    WHERE bookname = 'JAVA' ;
```



레코드 삭제하기

```
/* DELETE FROM 테이블명  
    WHERE 컬럼명=값 ; */
```

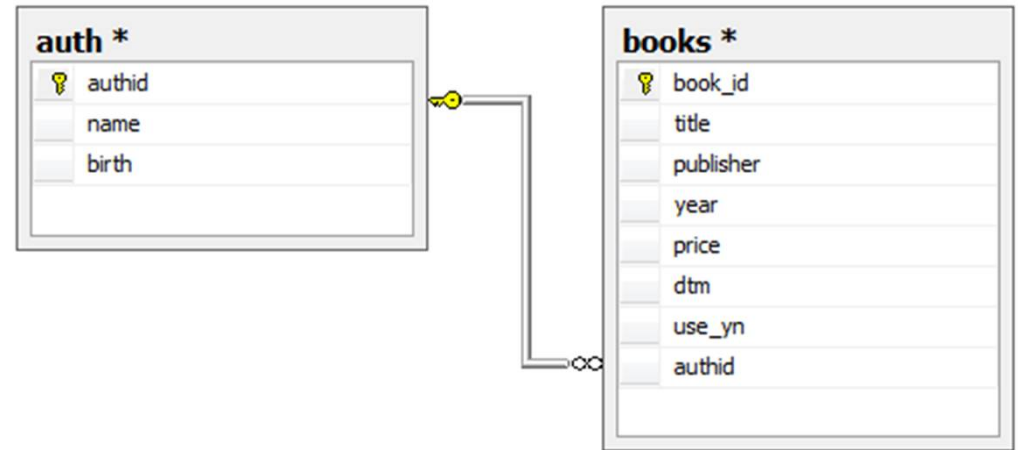
```
DELETE FROM book  
    WHERE bookname = 'JAVA' ;
```

```
DELETE FROM book  
    WHERE publisher ='Wiley' AND year < '2015' ;
```



레코드 조인

- 조인이란?
 - 테이블 연결



- 조인의 종류
 - *inner join*
 - *left join = left outer join*
 - *right join = right outer join*
 - *cross join*

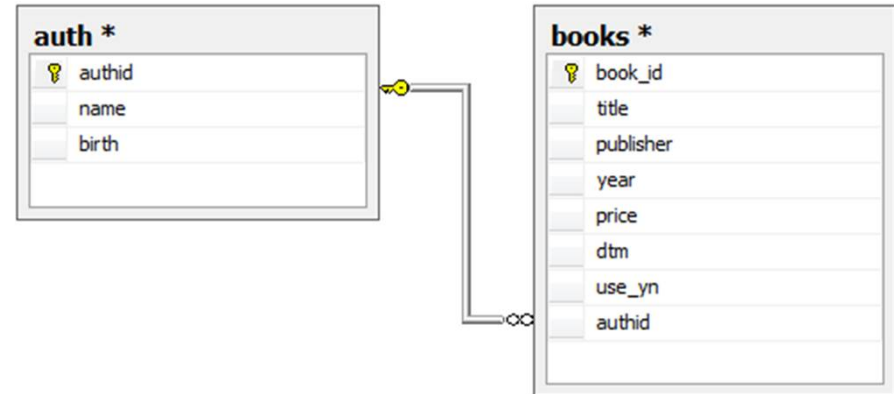


조인하기

-- *inner join*

```
select book.*, auth.*  
from book
```

```
inner join auth on book.authid = auth.authid ;
```



-- *left join*

```
select book.*, auth.*  
from book
```

```
left join auth on book.authid = auth.authid ;
```



JDBC 프로그래밍

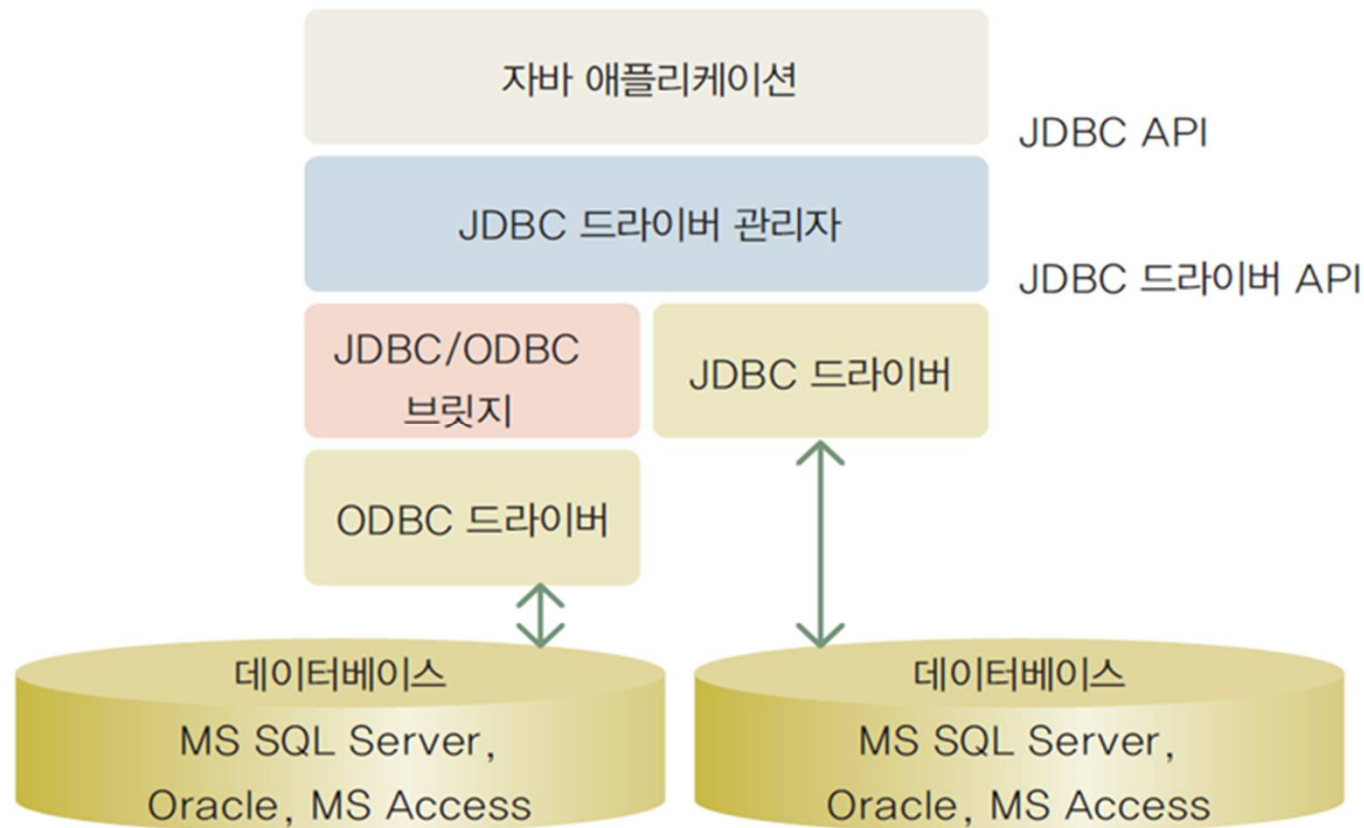
- 자바와 데이터베이스
- JDBC란
 - 자바에서 DB 프로그래밍을 위해 사용되는 라이브러리
- JDBC 프로그래밍 절차
 - JDBC 드라이버 로딩
 - 데이터베이스 커넥션 연결
 - PreparedStatement 객체 생성
 - 쿼리 실행 쿼리 실행 결과 사용
 - PreparedStatement 종료
 - 데이터베이스 커넥션 종료

자바를 통하여
데이터베이스
를 사용하는
방법을
학습합니다.



자바와 데이터베이스

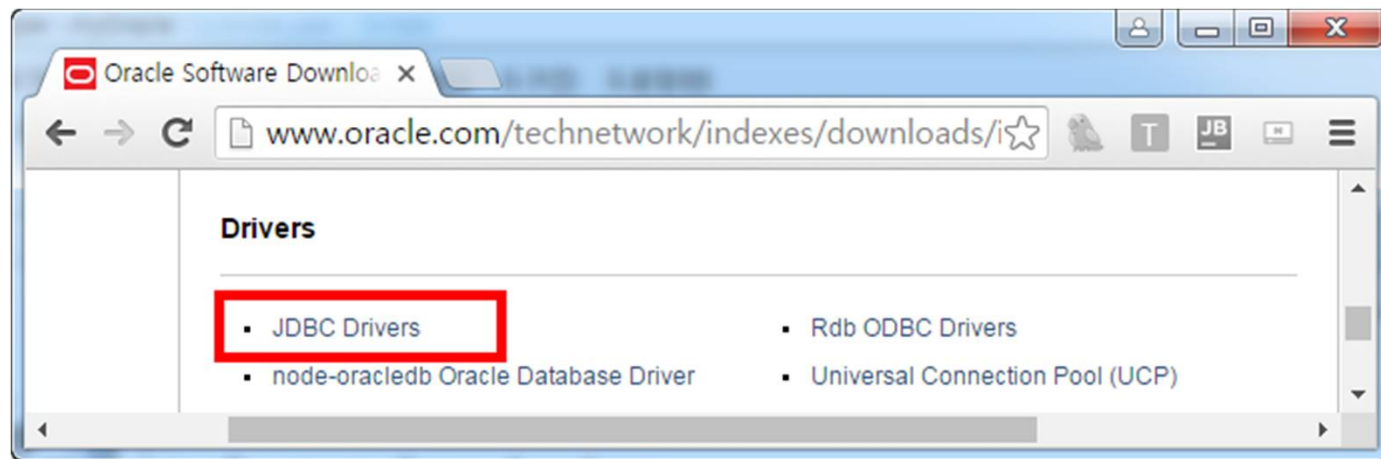
- JDBC(Java Database Connectivity)는 자바 API의 하나로서 데이터베이스에 연결하여서 데이터베이스 안의 데이터에 대하여 검색하고 데이터를 변경할 수 있게 한다.





JDBC 드라이버 다운로드

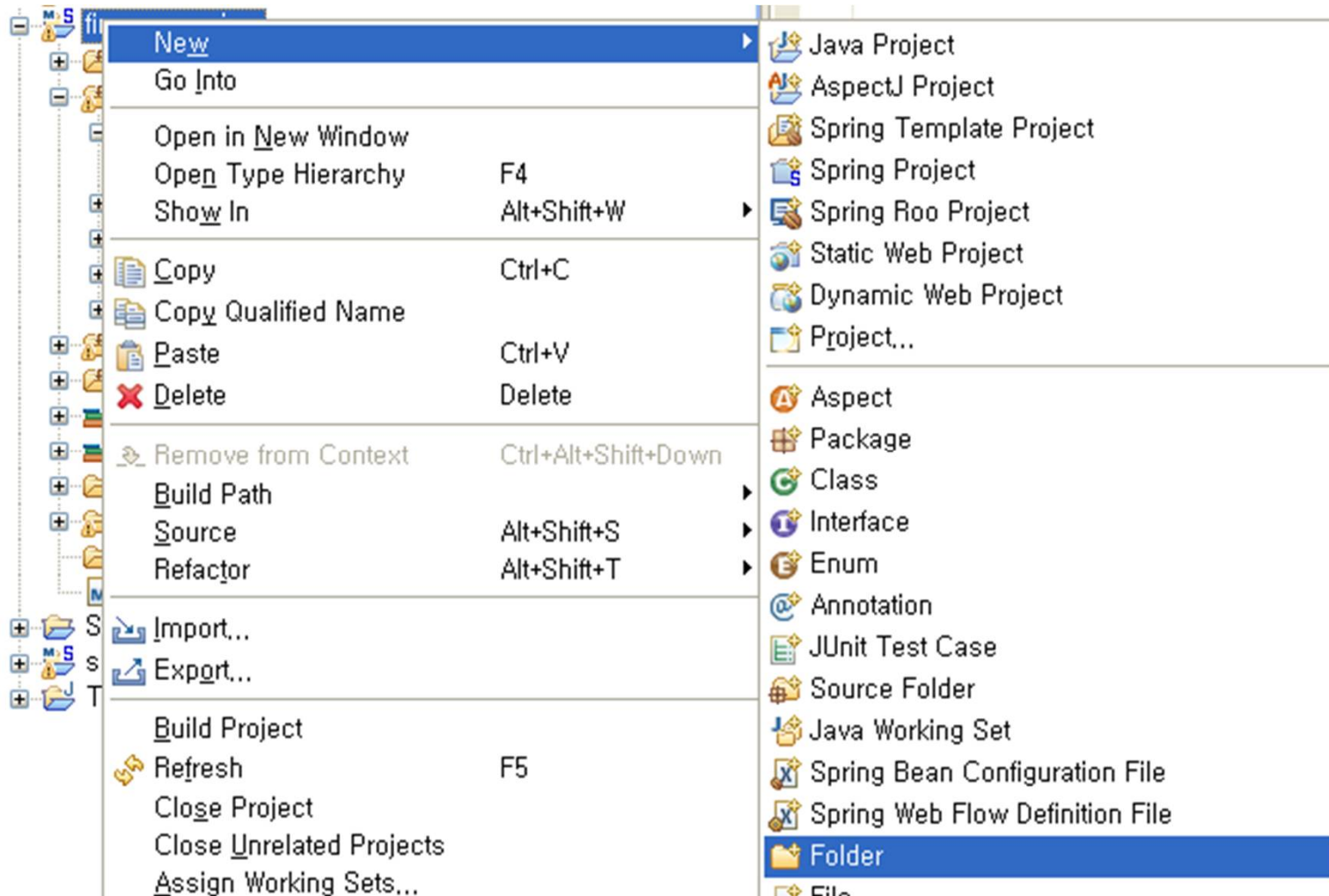
- Mysql / SQLite JDBC 드라이버 다운로드
 1. <http://mvnrepository.com/>에서 드라이버를 다운로드 받는다.
단, jar 파일이면 압축을 풀지 마시오.
- Oracle JDBC 드라이버 다운로드
 1. <https://www.oracle.com/downloads/>에서 드라이버를 다운로드 받는다.
단, jar 파일이면 압축을 풀지 마시오.





JDBC를 Build Path에 추가하기

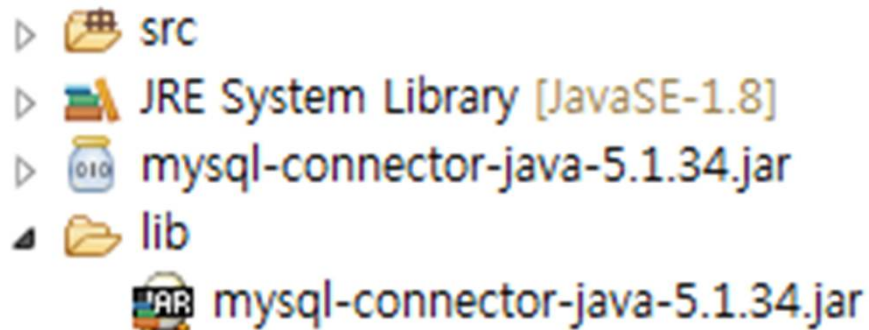
- 프로젝트에 lib 폴더 생성





JDBC를 Build Path에 추가하기

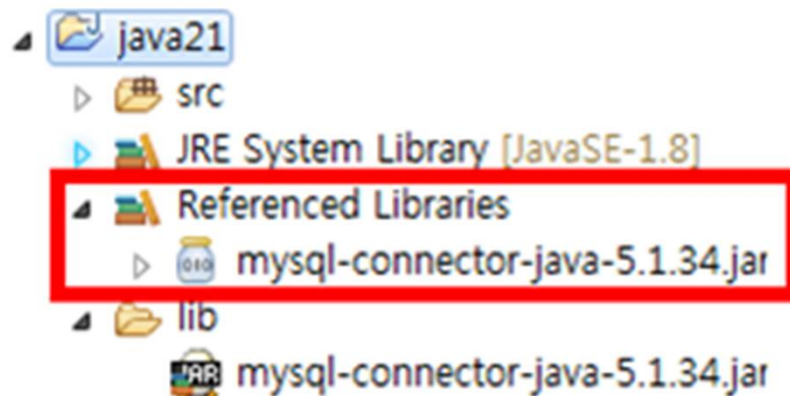
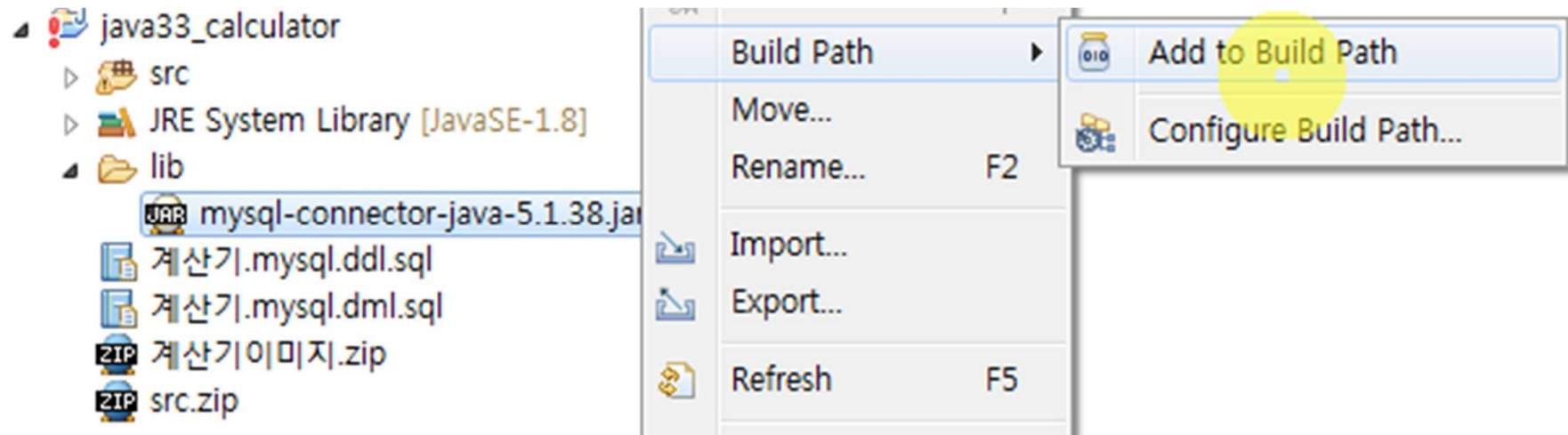
- lib 폴더에 다운로드 받은 jar 파일을 복사





JDBC를 Build Path에 추가하기

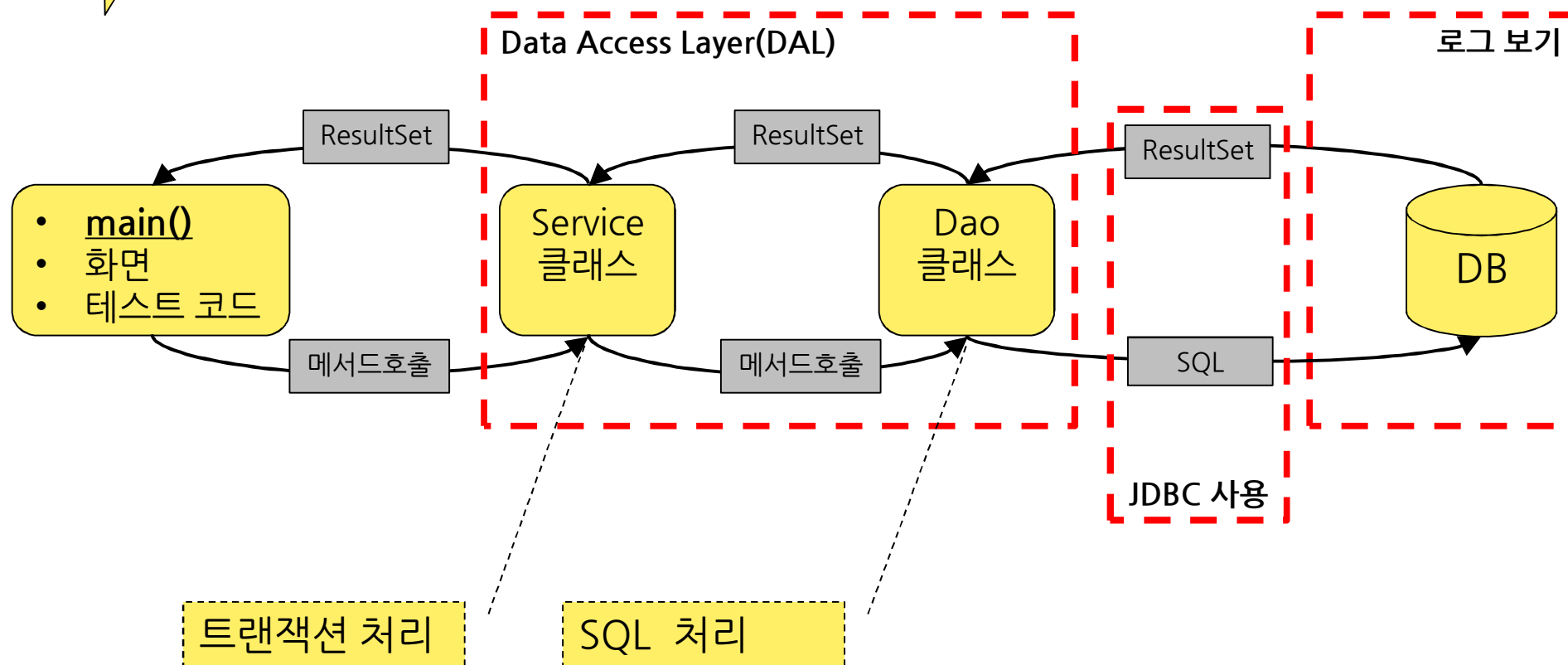
- jar 파일 Java Build Path에 추가





JDBC를 이용한 데이터 처리 과정

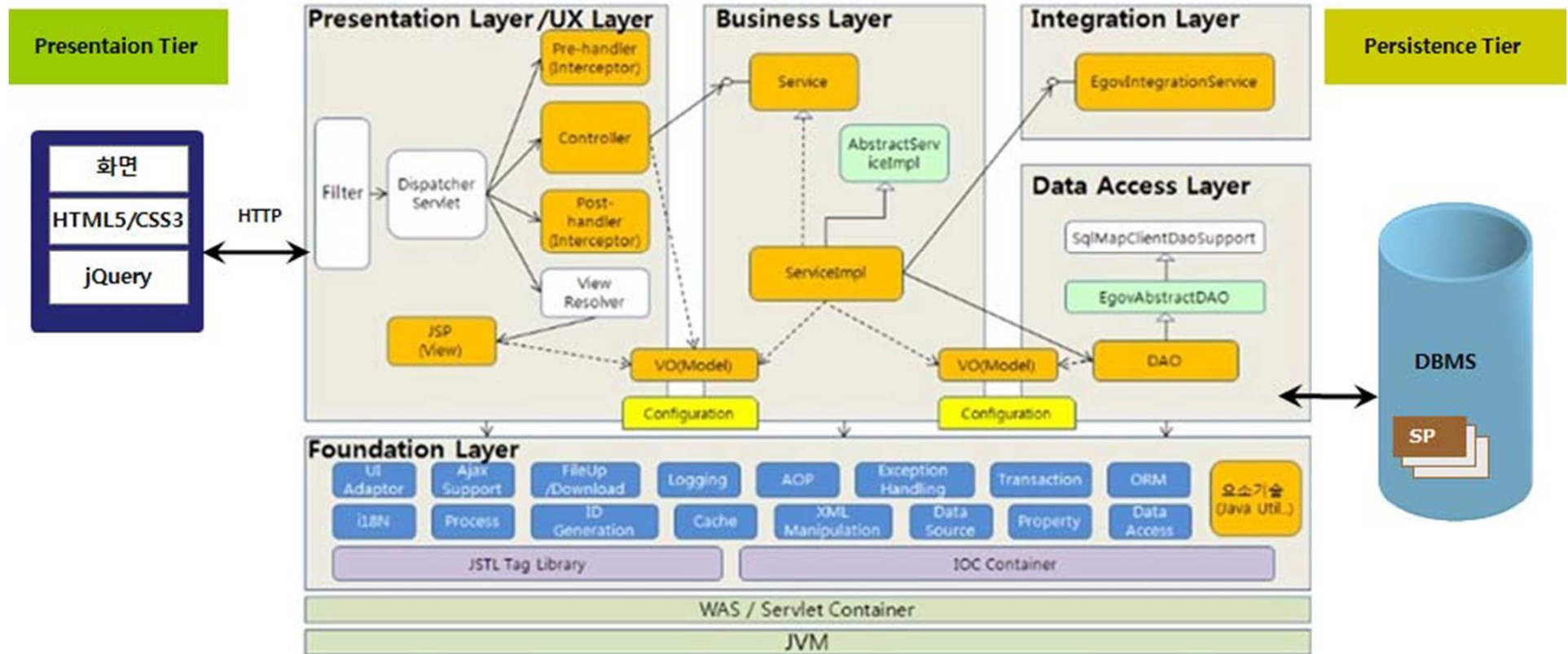
암기





전자 정부 프레임워크

- 전자정부 모바일 공통 컴포넌트 적용 아키텍처 예시



전자정부 프레임워크 v3.0



JDBC 프로그래밍 방식

- Statement 방식
 - *SQL 인젝션 공격에 취약*
 - *느리다*
- PreparedStatement 방식
 - *SQL 인젝션 공격을 무력화*
 - *빠르다*



JDBC 프로그래밍 방식

Application	DB I/O		DBMS						
GUI화면	Service Layer	DAO(Data Access Object) Layer	TABLE						
		<table><tr><td>SELECT</td><td>INSERT DELETE UPDATE</td></tr><tr><td>executeQuery()</td><td>executeUpdate()</td></tr><tr><td>ResultSet</td><td>int</td></tr></table>	SELECT	INSERT DELETE UPDATE	executeQuery()	executeUpdate()	ResultSet	int	
SELECT	INSERT DELETE UPDATE								
executeQuery()	executeUpdate()								
ResultSet	int								



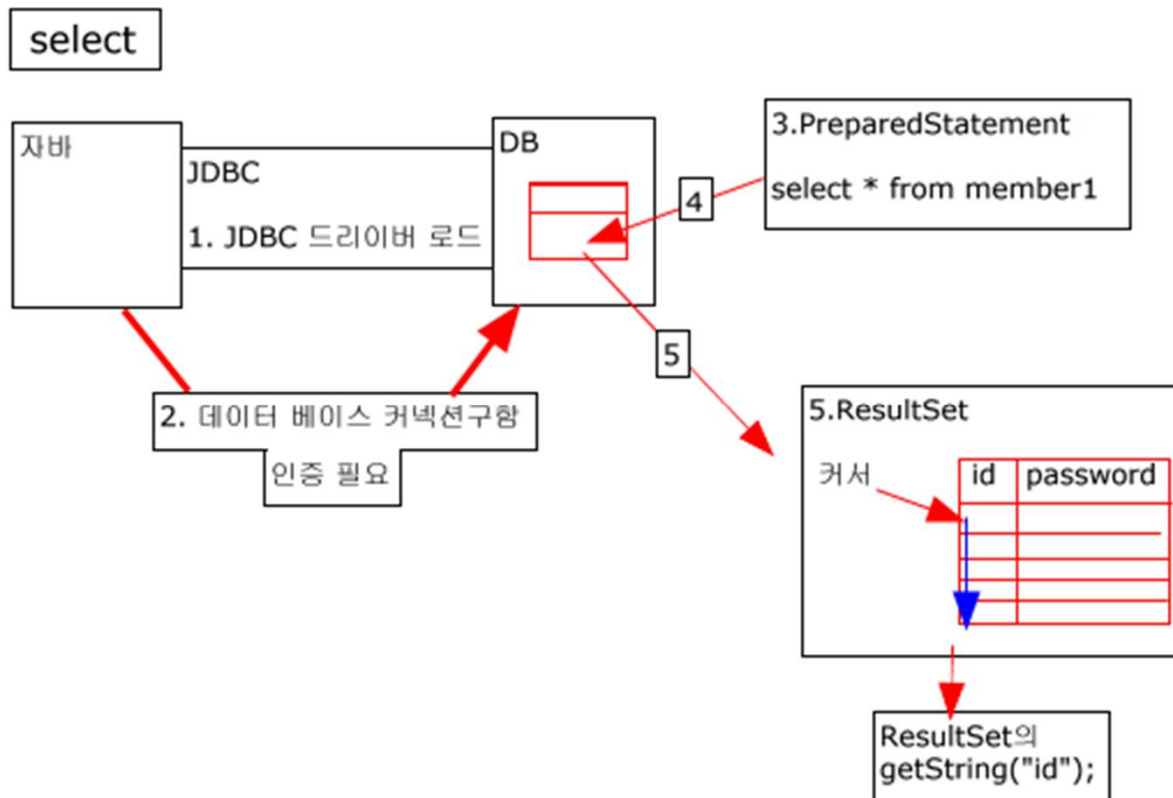
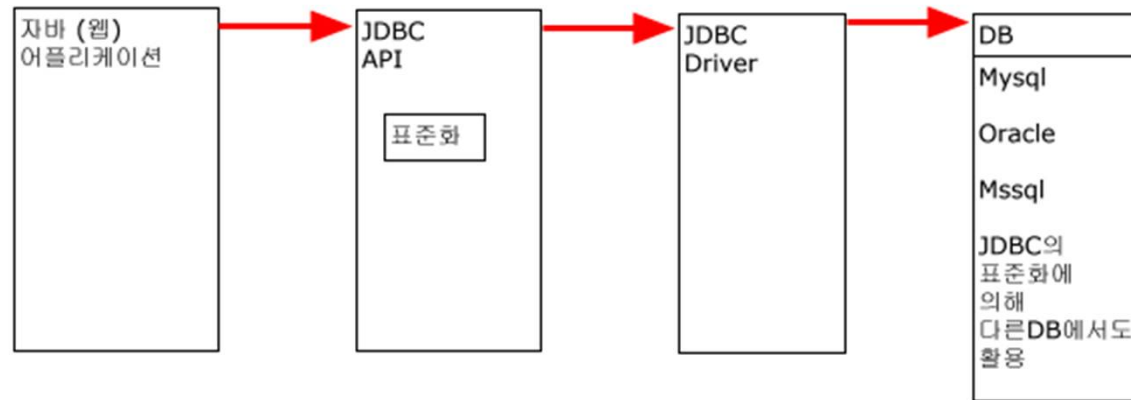
JDBC프로그램 실행 순서(1/3)

암기

1. JDBC 드라이버 로딩
2. 데이터베이스 커넥션 연결
3. 쿼리 실행을 위한 PreparedStatement 객체 생성
4. 쿼리 실행
 - executeQuery()
 - executeUpdate()
5. 쿼리 실행 결과 사용
6. PreparedStatement 종료
7. 데이터베이스 커넥션 종료

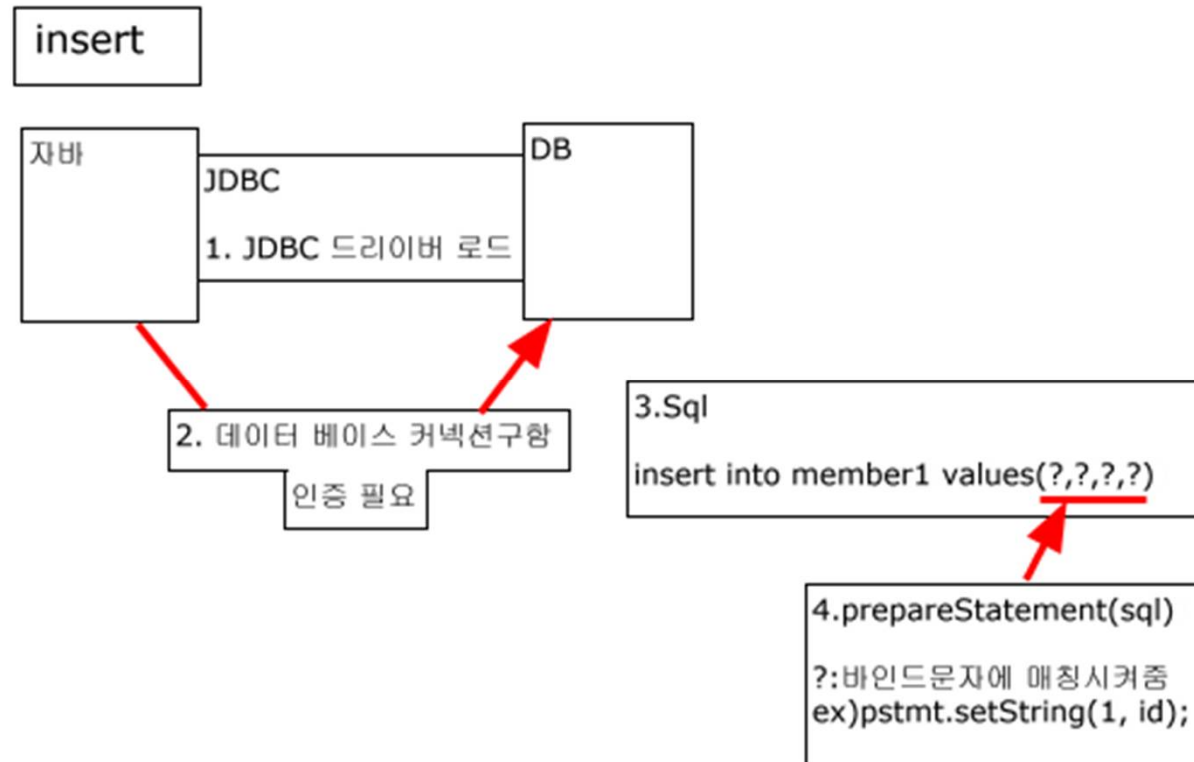
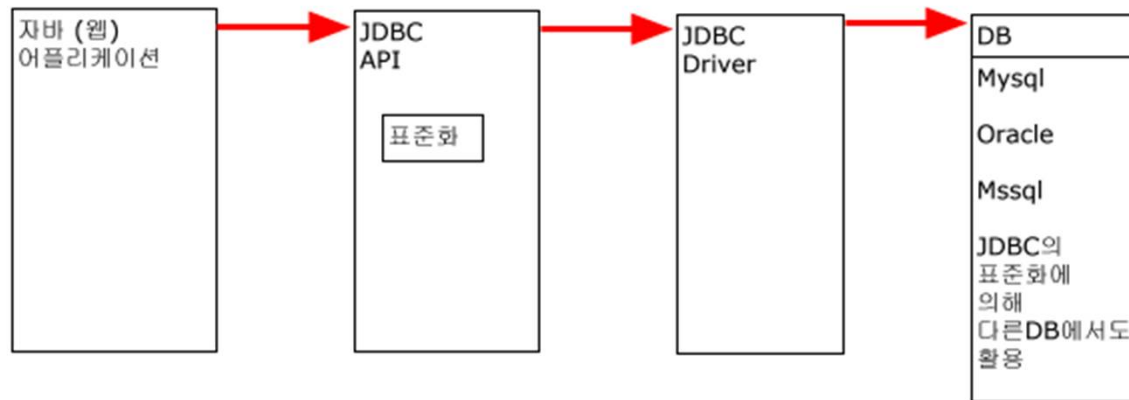


JDBC프로그램 실행 순서(2/3)





JDBC프로그램 실행 순서(3/3)





데이터베이스 프로그래밍의 순서

① JDBC 드라이버를 적재

② 데이터베이스 연결

```
Class.forName("      ?      ");
```

```
Connection conn =  
DriverManager.getConnection(url, user, pw);
```



③ SQL 문장 작성 및 전송

```
String query = "SELECT * FROM book ";  
PreparedStatement stmt =  
conn.prepareStatement(query);  
ResultSet rs = stmt.executeQuery();
```



④ 결과 집합 사용 후 연결 해제

```
while( rs.next ) {  
    int bookid      = rs.getInt( "bookid" );  
    String bookname = rs.getString( "bookname" );  
}
```





타입 매핑

sqlite(mysql, oracle)	jdbc 함수	java
BIT(1)	getBoolean() setBoolean()	boolean, Boolean
BIT(1), TINYINT, SMALLINT, INT, BIGINT	getInt() setInt()	byte, short, int, long
CHAR , VARCHAR TEXT	getString() setString()	String
FLOAT, DOUBLE	getDouble() setDouble()	float, double
NUMERIC, DECIMAL		java.math.BigDecimal java.math.BigInteger
DATE DATETIME TIME TIMESTAMP	getDate() setDate()	Java.util.Date Java.sql.Date Java.sql.Time Java.sql.Timestamp
BINARY, VARBINARY, BLOB		byte[]



DBMS별 타입 매핑

MySQL DBMS	Oracle DBMS	SQLite DBMS
INT	NUMBER(자리수, 0)	INTEGER
INTEGER		
TINYINT		
SMALLINT		
MEDIUMINT		
BIGINT		
NUMERIC		
BOOLEAN		
CHAR	VARCHAR2	TEXT
VARCHAR		
NCHAR		
NVARCHAR2		
TEXT		
DATE	DATE	TEXT
DATETIME		
BLOB	BLOB	NONE
CLOB	CLOB	
REAL	NUMBER(자리수, 소수점)	REAL
DOUBLE		
FLOAT		
DECIMAL(10,5)		



- DBConnect 클래스작성
 - java.jdbc.test.DBConnectTestJUnit 클래스 만들기
- Book 테이블 관련 클래스
 - java.jdbc.model.ModelBook 클래스 만들기
 - java.jdbc.infrc.IBook 인터페이스 만들기
 - java.jdbc.dao.DaoBook 클래스 만들기
 - java.jdbc.servcie.ServiceBook 클래스 만들기
 - java.jdbc.test.BookTestMain 클래스만들기 <-- 선택
 - java.jdbc.test.BookTestJUnit 클래스 만들기
- Auth 테이블 관련 클래스
 - java.jdbc.model.ModelAuth 클래스 만들기
 - java.jdbc.infrc.IAuth 인터페이스 만들기
 - java.jdbc.dao.DaoAuth 클래스 만들기
 - java.jdbc.servcie.ServiceAuth 클래스 만들기
 - java.jdbc.test.AuthTestJUnit 클래스 만들기



DBConnect 만들기 - MySQL용

```
public class DBConnect {  
    public static Connection connectionMySQL(){  
        String url      = "jdbc:mysql://localhost:3306/book_db";  
        String user      = "root";  
        String password = "root";  
        Connection conn = null;  
        try {  
            Class.forName("com.mysql.jdbc.Driver"); // driver 적재  
            conn = DriverManager.getConnection(url, user, password); // DB 연결  
        } catch (ClassNotFoundException e) {  
            e.printStackTrace();  
        } catch (SQLException e) {  
            e.printStackTrace();  
        }  
        return conn;  
    }  
    public static Connection makeConnection(){  
        return connectionOracle(); // Oracle과 연결할 때  
    }  
}
```




DBConnectTestJUnit 만들기

```
public class DBConnectTestJUnit {  
    @Test  
    public void connectionMySQL() throws Exception {  
        Connection conn = DBConnect.connectionMySQL();  
  
        if( conn != null) {  
            assertTrue("db connect success", true);  
        }  
        else {  
            assertTrue("db connect fail", false);  
        }  
    }  
  
    @Test  
    public void makeConnection() throws Exception {  
        Connection conn = DBConnect.makeConnection();  
  
        if( conn != null) {  
            assertTrue("db connect success", true);  
        }  
        else {  
            assertTrue("db connect fail", false);  
        }  
    }  
}
```



ModelBook 만들기

```
public class ModelBook {  
    private Integer bookid    = null;  
    private String  bookname  = "" ;  
    private String  publisher = "" ;  
    private String  year      = "" ;  
    private Integer price     = null;  
    private Date    dtm       = null;  
    private Boolean use_yn     = null;  
    private Integer authid    = null;  
  
    // getter & setter 만들기  
  
    // 디폴트 생성자(constructor) 만들기  
  
    // toString() 만들기  
  
}
```



IBook 만들기

```
public interface IBook {  
  
    int getCount(ModelBook book) throws SQLException;  
  
    int getMaxBookid() throws SQLException;  
  
    ResultSet selectAll() throws SQLException;  
  
    ResultSet selectLike(ModelBook book) throws SQLException;  
  
    ResultSet selectEqual(ModelBook book) throws SQLException;  
  
    ResultSet selectDynamic(ModelBook book) throws SQLException;  
  
    int insertBook(ModelBook book) throws SQLException;  
  
    int updateBook(ModelBook wherebook, ModelBook setbook) throws SQLException;  
  
    int deleteBook(ModelBook book) throws SQLException;  
}
```



DaoBook 만들기

```
public class DaoBook implements IBook {  
    private Connection conn = null;  
    public DaoBook(Connection conn) { this.conn = conn; }  
  
    @Override  
    public int getCount(ModelBook book) { return -1; }  
    @Override  
    public int getMaxBookid() { return -1; }  
    @Override  
    public ResultSet selectAll() { return null; }  
    @Override  
    public ResultSet selectLike(ModelBook book) { return null; }  
    @Override  
    public ResultSet selectEqual(ModelBook book) { return null; }  
    @Override  
    public int insertBook(ModelBook book) { return 0; }  
    @Override  
    public int updateBook(ModelBook wherebook, ModelBook setbook) { return 0; }  
    @Override  
    public int deleteBook(ModelBook book) { return 0; }  
    @Override  
    public ResultSet selectDynamic(ModelBook book) { return null; }  
}
```



DaoBook의 getCount(...) 메서드

```
public int getCount(ModelBook book) throws SQLException {
    int result = -1;

    // SQL 문장
    String query = "SELECT count(*) as total from book where 1 = 1 ";

    try {
        // 문장 객체 생성
        PreparedStatement stmt = conn.prepareStatement(query);

        // SQL 문장 실행
        ResultSet rs = stmt.executeQuery();

        rs.next(); // 커서 이동. ResultSet 첫번째로 로우로.
        result = rs.getInt("total"); // total 컬럼의 값을 가져온다.
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return result;
}
```

SQL문장을 실행하고
결과로 **ResultSet**을 반환한다.



DaoBook의 getMaxBookid(...) 메서드

```
public int getMaxBookid() throws SQLException {
    int result = -1;

    // SQL 문장
    String query = " select max(bookid) maxid from book ";

    try {
        // 문장 객체 생성
        PreparedStatement stmt = conn.prepareStatement(query);

        // SQL 문장 실행
        ResultSet rs = stmt.executeQuery();

        rs.next(); // 커서 이동. ResultSet 첫번째로 로우로.
        result = rs.getInt("maxid"); // maxid 컬럼의 값을 가져온다.
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return result;
}
```

SQL문장을 실행하고
결과로 **ResultSet**을 반환한다.



DaoBook의 selectAll(...) 메서드

```
public ResultSet selectAll() {  
    ResultSet rs = null;  
  
    // SQL 문장 실행  
    String query = "SELECT * FROM book ORDER BY bookid ASC ";  
  
    try {  
        // 문장 객체 생성  
        PreparedStatement stmt = conn.prepareStatement(query);  
  
        // SQL 문장 실행  
        rs = stmt.executeQuery();  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
  
    return rs;  
}
```

SQL문장을 실행하고
결과로 **ResultSet**을 반환한다.



DaoBook의 selectLike(...) 메서드

```
public ResultSet selectLike(String bookname) {  
  
    ResultSet rs = null;  
  
    try {  
        // SQL 문장 생성  
        String query = "SELECT * FROM book where bookname like ? ";  
  
        // 문장 객체 생성  
        PreparedStatement stmt = conn.prepareStatement(query);  
        stmt.setString(1, "%" + book.getBookname() + "%");  
  
        // SQL 문장 실행  
        rs = stmt.executeQuery();  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
  
    return rs;  
}
```

SQL문장을 실행하고
결과 row 를 반환한다.



DaoBook의 selectEqual(..) 메서드

```
public ResultSet selectEqual(ModelBook book) {  
  
    ResultSet rs = null;  
  
    try {  
        // SQL 문장 생성  
        String query = "SELECT * FROM book where bookname = ? ";  
  
        // 문장 객체 생성  
        PreparedStatement stmt = conn.prepareStatement(query);  
        stmt.setString(1, book.getBookname());  
  
        // SQL 문장 실행  
        rs = stmt.executeQuery();  
  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
  
    return rs;  
}
```

SQL문장을 실행하고
결과 row 를 반환한다.



DaoBook의 insertBook(...) 메서드

```
public int insertBook(ModelBook book) throws SQLException {  
    int rs = 0;  
    try {  
        String query = " INSERT INTO ";  
        query += " BOOK( BOOKNAME, PUBLISHER, YEAR, PRICE, DTM, USE_YN, AUTHID) ";  
        query += " VALUES( ?, ?, ?, ?, ?, ?, ? ) ";  
        PreparedStatement stmt = conn.prepareStatement(query);  
        stmt.setString (1, book.getBookname()           );  
        stmt.setString (2, book.getPublisher()           );  
        stmt.setString (3, book.getYear()                 );  
        stmt.setInt     (4, book.getPrice()               );  
        stmt.setDate    (5, (java.sql.Date) book.getDtm());  
        stmt.setBoolean(6, book.getUse_yn()               );  
        stmt.setInt     (7, book.getAuthid()              );  
  
        rs = stmt.executeUpdate();  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
    return rs;  
}
```

SQL문장을 실행하고
영향 받은 row의 갯수를 반환한다.



DaoBook의 updateBook(...) 메서드

```
public int updateBook(ModelBook wherebook, ModelBook setbook) {  
    int rs = 0;  
    try {  
        // SQL 문장 생성  
        String query = "UPDATE book\n";  
        query      += "    SET year    = ?, price    = ?\n";  
        query      += " WHERE bookname = ?\n";  
  
        // 문장 객체 생성  
        PreparedStatement stmt = conn.prepareStatement(query);  
        stmt.setString ( 1, setbook.getYear()      );  
        stmt.setInt    ( 2, setbook.getPrice()     );  
        stmt.setString ( 3, wherebook.getBookname() );  
  
        // SQL 문장 실행  
        rs = stmt.executeUpdate();  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
    return rs;  
}
```

SQL문장을 실행하고
영향 받은 **row**의 갯수를 반환한다.



DaoBook의 deleteBook(...) 메서드

```
public int deleteBook(String bookname) {  
  
    int rs = 0;  
  
    try {  
        // SQL 문장 생성  
        String query = "delete from book where bookname = ? ";  
  
        // 문장 객체 생성  
        PreparedStatement stmt = conn.prepareStatement(query);  
        stmt.setString ( 1, book.getBookname() );  
  
        // SQL 문장 실행  
        rs = stmt.executeUpdate();  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
  
    return rs;  
}
```

SQL문장을 실행하고
영향 받은 **row**의 갯수를 반환한다.



DaoBook의 selectDynamic(..) 메서드

```
public ResultSet selectDynamic(ModelBook book) throws SQLException {
    ResultSet result = null;

    try {
        // query 작성
        String query = " select * from book  \n";
        query += "  where 1 = 1          \n";
        if( book.getBookid() != null      ) query += "    and bookid    = ?  \n";
        if( !book.getBookname().isEmpty() ) query += "    and bookname = ?  \n";

        // 문장 객체 생성
        PreparedStatement stmt = conn.prepareStatement(query);
        int c = 1;
        if( book.getBookid() != null      ) stmt.setInt    ( c++, book.getBookid() );
        if( !book.getBookname().isEmpty() ) stmt.setString( c++, book.getBookname());

        // query 실행
        result = stmt.executeQuery();
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return result;
}
```



ServiceBook 만들기

```
public class ServiceBook implements IBook {  
    private Connection conn = null;  
    public ServiceBook() { this.conn = DBConnect.makeConnection(); }  
  
    @Override public int getCount(ModelBook book) { return -1; }  
    @Override public int getMaxBookid() { return -1; }  
    @Override public ResultSet selectAll() { return null; }  
    @Override public ResultSet selectLike(ModelBook book) { return null; }  
    @Override public ResultSet selectEqual(ModelBook book) { return null; }  
    @Override public int insertBook(ModelBook book) { return -1; }  
    @Override public int updateBook(ModelBook wherebook, ModelBook setbook) {  
        return -1;  
    }  
    @Override public int deleteBook(ModelBook book) { return -1; }  
    @Override public ResultSet selectDynamic(ModelBook book) { return null; }  
  
    public int transCommit(ModelBook b1, ModelBook b2) { return null; }  
    public int transRollback(ModelBook b1, ModelBook b2) { return null; }  
}
```



ServiceBook의 getCount(...) 메서드

```
@Override
public int getCount(ModelBook book) throws SQLException {

    int rs = -1;

    try {
        // 트랜잭션 시작
        conn.setAutoCommit( false );
        DaoBook dao = new DaoBook( conn );
        rs= dao.getCount(book);
        conn.commit();
    } catch (SQLException e) {
        e.printStackTrace();
        conn.rollback();
    }

    return rs;
}
```



ServiceBook의 getMaxBookid(...) 메서드

```
@Override
public int getMaxBookid() throws SQLException {

    int rs = -1;

    try {
        // 트랜잭션 시작
        conn.setAutoCommit( false );
        DaoBook dao = new DaoBook( conn );
        rs = dao.getMaxBookid();
        conn.commit();
    } catch (SQLException e) {
        e.printStackTrace();
        conn.rollback();
    }

    return rs;
}
```




ServiceBook의 selectAll(...) 메서드

```
@Override
public ResultSet selectAll() throws SQLException {

    ResultSet rs = null;

    try {
        // 트랜잭션 시작
        conn.setAutoCommit( false );
        DaoBook dao = new DaoBook( conn );
        rs = dao.selectAll();
        conn.commit();
    } catch (SQLException e) {
        e.printStackTrace();
        conn.rollback();
    }

    return rs;
}
```



ServiceBook의 selectLike(...) 메서드

```
@Override
public ResultSet selectLike(ModelBook book) throws SQLException {

    ResultSet rs = null;

    try {
        // 트랜잭션 시작
        conn.setAutoCommit( false );
        DaoBook dao = new DaoBook(conn);
        rs = dao.selectLike(book);
        conn.commit();
    } catch (SQLException e) {
        e.printStackTrace();
        conn.rollback();
    }
    return rs;
}
```



ServiceBook의 selectEqual(...) 메서드

```
@Override
public ResultSet selectEqual(ModelBook book) throws SQLException {

    ResultSet rs = null;

    try {
        // 트랜잭션 시작
        conn.setAutoCommit( false );
        DaoBook dao = new DaoBook(conn);
        rs = dao.selectEqual(book);
        conn.commit();
    } catch (SQLException e) {
        e.printStackTrace();
        conn.rollback();
    }
    return rs;
}
```



ServiceBook의 insertBook(...) 메서드

```
@Override
public int insertBook(ModelBook book) throws SQLException {

    int result = -1;

    try {
        // 트랜잭션 시작
        conn.setAutoCommit(false);
        DaoBook dao = new DaoBook(conn);
        result = dao.insertBook(book);
        conn.commit();
    } catch (SQLException e) {
        e.printStackTrace();
        conn.rollback();
    }

    return result;
}
```



ServiceBook의 updateBook(...) 메서드

```
@Override
public int updateBook(ModelBook wherebook, ModelBook setbook) throws SQLException {

    int result = -1;

    try {
        // 트랜잭션 시작
        conn.setAutoCommit(false);
        DaoBook dao = new DaoBook(conn);
        result = dao.updateBook(wherebook, setbook);
        conn.commit();
    } catch (SQLException e) {
        e.printStackTrace();
        conn.rollback();
    }

    return result;
}
```



ServiceBook의 deleteBook(...) 메서드

```
@Override
public int deleteBook(ModelBook book) throws SQLException {

    int result = -1;

    try {
        // 트랜잭션 시작
        conn.setAutoCommit(false);
        DaoBook dao = new DaoBook(conn);
        result = dao.deleteBook(book);
        conn.commit();
    } catch (SQLException e) {
        e.printStackTrace();
        conn.rollback();
    }

    return result;
}
```



ServiceBook의 selectDynamic(...) 메서드

```
@Override
public ResultSet selectDynamic(ModelBook book) throws SQLException {

    ResultSet rs = null;

    try {
        // 트랜잭션 시작
        conn.setAutoCommit( false );
        DaoBook dao = new DaoBook(conn);
        rs = dao.selectDynamic(book);
        conn.commit();
    } catch (SQLException e) {
        e.printStackTrace();
        conn.rollback();
    }
    return rs;
}
```



ServiceBook의 transCommit(...) 메서드

```
public int transCommit(ModelAuth auth, ModelBook book) throws SQLException {  
    int result = -1;  
  
    try {  
        conn.setAutoCommit(false);  
        DaoAuth daoAuth = new DaoAuth(conn);  
        result = daoAuth.insert(auth);  
  
        // 추가. Auth 테이블에 inserted 된 authid 를  
        // book 테이블에 insert 할 때 사용하도록 수정.  
        book.setAuthid( result );  
  
        DaoBook daoBook = new DaoBook(conn);  
        result = daoBook.insert(book );  
        conn.commit();  
    } catch (SQLException e) {  
        e.printStackTrace();  
        conn.rollback();  
    }  
  
    return result;  
}
```




ServiceBook의 transRollback(...) 메서드

```
public ResultSet transRollback(ModelBook b1, ModelBook b2) throws SQLException {  
  
    ResultSet rs = null;  
  
    try {  
        // 트랜잭션 시작  
        conn.setAutoCommit(false);  
        DaoBook dao = new DaoBook(conn);  
        dao.deleteBook(b1);  
        dao.insertBook(b2);  
        rs = dao.selectAll();  
        throw new SQLException();  
    } catch (SQLException e) {  
        e.printStackTrace();  
        conn.rollback();  
    }  
    return rs;  
}
```



ServiceBook에 printResultSet(...) 만들기

1. ResultSet 을 매개변수로 받아 출력하는 메서드, printResultSet 메서드를 만드시오. (출력 예시)

1	operating system	wiley	2003	30700	2004-01-01	false
2	mysql	oreilly	2009	58700	2010-01-01	true
3	java	hall	2013	40000	2014-01-01	true
4	first sql	wiley	2015	57700	2016-01-01	true

2. selectAll(), selectLike(), selectEqual()에서 ResultSet 출력 부분을 삭제하시오.
3. main 메서드에서 selectAll() 호출 시 결과를 반환 받아 결과를 출력하도록 하시오.

```
rs = dao.selectAll(conn);  
dao.printResultSet(rs);
```
4. main 메서드에서 selectLike() 호출 시 결과를 반환 받아 결과를 출력하도록 하시오.
5. main 메서드에서 selectEqual() 호출 시 결과를 반환 받아 결과를 출력하도록 하시오.



ServiceBook의 printResultSet(...) 메서드

```
public void printResultSet( ResultSet rs){
    rs.first();
    try {
        for ( ; rs.next(); ) {
            int id          = rs.getInt    ("bookid"    );
            String bookname  = rs.getString ("bookname" );
            String publisher = rs.getString ("publisher" );
            String year      = rs.getString ("year"      );
            int price        = rs.getInt    ("price"      );
            Date dtm         = rs.getDate   ("dtm"         );
            boolean use_yn   = rs.getBoolean("use_yn"     );

            String out = String.format(" %d \t %15s \t %10s \t %4s \t %7d \t %10s \t %b", id, bookname, publisher, year, price, dtm, use_yn);
            System.out.println(out);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```



BookTestJUnit 만들기

```
public class DaoBookTestJUnit {  
  
    private static ServiceBook svr = null;  
  
    @BeforeClass  
    public static void setClass() throws Exception {  
        svr = new ServiceBook();  
    }  
  
    @Test public void selectAll() throws Exception { }  
    @Test public void selectLike() throws Exception { }  
    @Test public void selectEqual() throws Exception { }  
    @Test public void insertBook() throws Exception { }  
    @Test public void updateBook() throws Exception { }  
    @Test public void deleteBook() throws Exception { }  
  
    @Test public void selectDynamic() throws Exception { }  
    @Test public void transCommit() throws Exception { }  
    @Test public void transRollback() throws Exception { }  
}
```



BookTestJUnit의 selectAll() 테스트 코드 작성

```
public class BookTestJUnit {
    static ServiceBook svr = null;

    @BeforeClass
    public static void setClass() throws Exception {
        svr = new ServiceBook();
    }

    @Test
    public void selectAll() throws Exception {
        ModelBook book = new ModelBook();

        ResultSet rs = svr.selectAll();
        assertNotNull(rs);
        rs.last();
        assertEquals(rs.getRow(), svr.getCount() );
    }
}
```



BookTestJUnit의 selectLike() 테스트 코드 작성

```
@Test
public void selectLike() throws Exception {

    ModelBook book = new ModelBook();
    book.setBookname("java");

    ResultSet rs = svr.selectLike(book);
    assertNotNull( rs );
    rs.last();

    ResultSet rs2 = svr.selectAll(book);
    assertNotNull(rs2);
    int count = 0;
    for( ; rs2.next() ; ) {
        if( rs2.getString("bookname").contains( book.getBookname() )){
            count = count +1;
        }
    }
    assertEquals( rs.getRow(), count );
}
```



BookTestJUnit의 selectEqual() 테스트 코드 작성

```
@Test
public void selectEqual() throws Exception {
    ModelBook book = new ModelBook();
    book.setBookname("java");

    ResultSet rs = svr.selectEqual(book);
    assertNotNull( rs );
    rs.last();

    ResultSet rs2 = svr.selectAll(book);
    assertNotNull(rs2);
    int count = 0;
    for( ; rs2.next() ; ) {
        if( rs2.getString("bookname").equalsIgnoreCase( book.getBookname() ) ){
            count = count +1;
        }
    }
    assertEquals( rs.getRow(), count );
}
```



BookTestJUnit의 insertBook() 테스트 코드 작성

```
@Test
public void insert() throws Exception {
    ModelBook book = new ModelBook();
    book.setBookname("");
    book.setAuthid( 1 );
    book.setDtm(Date.valueOf("2017-01-01"));
    book.setPrice(2000);
    //book.setBookid(1000);
    book.setYear("2017");
    book.setUse_yn(true);

    int rs = svr.insert(book);

    if( rs >=1 )
        assertTrue(true);
    else
        assertTrue(false);
}
```




BookTestJUnit의 updateBook() 테스트 코드 작성

```
@Test
public void update() throws Exception {

    ResultSet rs = svr.selectAll(null);
    assertNotNull(rs);
    rs.next();

    ModelBook where = new ModelBook();
    where.setBookname(rs.getString("bookname"));

    ModelBook set = new ModelBook();
    set.setYear("2017");
    set.setPrice(100);

    int rlt = svr.update(where, set);

    if (rlt >= 0)
        assertTrue(true);
    else
        assertTrue(false);
}
```



BookTestJUnit의 deleteBook() 테스트 코드 작성

```
@Test
public void delete() throws Exception {
    ModelBook book = new ModelBook();
    book.setBookname("aaa");

    int rlt = svr.delete(book);

    if (rlt >= 0) {
        assertTrue(true);
    }
    else {
        assertTrue(false);
    }
}
```



BookTestJUnit의 selectDynamic_1() 테스트 코드 작성

```
@Test
public void selectDynamic_1() throws Exception {
    ModelBook book = new ModelBook();
    // select * from book where 1 = 1
    ResultSet rs = svr.selectDynamic(book);
    assertNotNull(rs);
    rs.last();

    assertEquals( rs.getRow(), svr.getCount(null) );
}
```



BookTestJUnit의 selectDynamic_2() 테스트 코드 작성

```
@Test
public void selectDynamic_2() throws Exception {

    // select * from book where 1 = 1 and bookid = 1;
    ResultSet rs2 = svr.selectAll(null);
    assertNotNull(rs2);
    rs2.next();

    ModelBook book = new ModelBook();
    book.setBookid( rs2.getInt("bookid") );

    ResultSet rs = svr.selectDynamic(book);
    assertNotNull(rs);
    rs.next();

    assertEquals( rs.getString("bookname"), rs2.getString("bookname") );
}
```



BookTestJUnit의 selectDynamic_3() 테스트 코드 작성

```
@Test
public void selectDynamic_3() throws Exception {

    // select * from book where 1 = 1 and bookname= 'aaa';
    ResultSet rs2 = svr.selectAll(null);
    assertNotNull(rs2);
    rs2.next();

    ModelBook book = new ModelBook();
    book.setBookname( rs2.getString("bookname") );

    ResultSet rs = svr.selectDynamic(book);
    assertNotNull(rs);
    rs.next();

    ResultSet rs3 = svr.selectEqual(book);
    assertNotNull(rs3);
    rs3.next();

    assertEquals( rs.getString("bookname"), rs3.getString("bookname") );
}
```



BookTestJUnit의 selectDynamic_4() 테스트 코드 작성

```
@Test
public void selectDynamic_4() throws Exception {

    // select * from book where 1 = 1 and bookid = 1 and bookname= 'aaa';
    ResultSet rs2 = svr.selectAll(null);
    assertNotNull(rs2);
    rs2.next();

    ModelBook book = new ModelBook();
    book.setBookid ( rs2.getInt("bookid") );
    book.setBookname( rs2.getString("bookname") );

    ResultSet rs = svr.selectDynamic(book);
    assertNotNull(rs);
    rs.next();

    ResultSet rs3 = svr.selectEqual(book);
    assertNotNull(rs3);
    rs3.next();

    assertEquals( rs.getString("bookname"), rs3.getString("bookname") );
}
```



BookTestJUnit의 transCommit() 테스트 코드 작성

```
@Test
public void transCommit() throws Exception {
    ModelAuth auth = new ModelAuth();
    auth.setName( " test auth");
    auth.setBirth("1910-01-01");

    ModelBook book = new ModelBook();
    book.setBookname("test book");
    book.setUse_yn(true);
    book.setPublisher("test pub");
    book.setPrice(20000);
    book.setYear("2011");
    book.setDtm(java.sql.Date.valueOf("2017-05-15"));

    int result = svr.transCommit(auth, book);

    if(result>=0 )
        assertTrue(true);
    else
        assertTrue(false);
}
```



BookTestJUnit의 transRollback() 테스트 코드 작성

```
@Test  
public void transRollback() throws Exception {  
  
}
```