

# 삼성 청년 SW 아카데미

JDBC

## <알림>

본 강의는 삼성 청년 SW아카데미의 콘텐츠로  
보안서약서에 의거하여  
강의 내용을 어떠한 사유로도 임의로 복사,  
촬영, 녹음, 복제, 보관, 전송하거나  
허가 받지 않은 저장매체를  
이용한 보관, 제3자에게 누설, 공개,  
또는 사용하는 등의 행위를 금합니다.

## 오늘의 포인트

- DML (INSERT, UPDATE, DELETE)
- SELECT
- JDBC

**Table**

# Table

✓ Table.

Confidential

열 (Column)

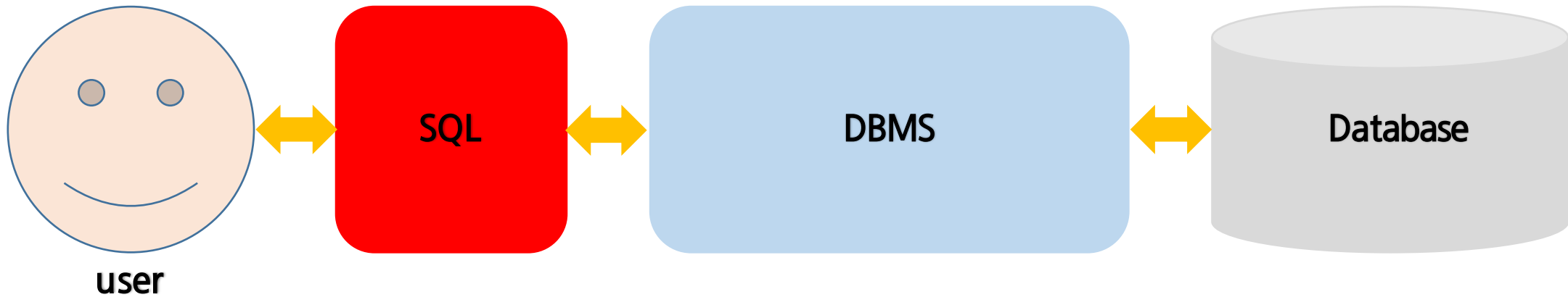
Column Name

employee_id	first_name	last_name	email	phone_number	hire_date	job_id	salary	commission_pct	manager_id	department_id
100	Steven	King	SKING	515.123.4567	1987-10-17	AD_PRES	24000.00	NULL	NULL	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	1989-11-21	AD_VP	17000.00	NULL	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	1993-04-13	AD_VP	17000.00	NULL	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	1990-01-03	IT_PROG	9000.00	NULL	102	60
104	Bruce	Ernst	BERNST	590.423.4568	1991-05-21	IT_PROG	6000.00	NULL	103	60
105	David	Austin	DAUSTIN	590.423.4569	1997-06-25	IT_PROG	4800.00	NULL	103	60
106	Valli	Pataballa	VPATABAL	590.423.4560	1998-09-05	IT_PROG	4800.00	NULL	103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	1999-09-07	IT_PROG	4200.00	NULL	103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	1994-11-17	FI_MGR	12000.00	NULL	101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	1994-10-16	FI_ACCOUNT	9000.00	NULL	108	100
110	John	Chen	JCHEN	515.124.4269	1997-12-28	FI_ACCOUNT	8200.00	NULL	108	100
111	Ismael	Sciarra	ISCIARRA	515.124.4369	1997-10-30	FI_ACCOUNT	7700.00	NULL	108	100
112	Jose Manuel	Urman	JMURMAN	515.124.4469	1998-05-07	FI_ACCOUNT	7800.00	NULL	108	100

행 (Row)

### ✓ SQL (Structured Query Language)

- Database에 있는 정보를 사용할 수 있도록 지원하는 언어.
- 모든 DBMS에서 사용 가능.
- 대소문자는 구별하지 않음 (단, 데이터의 대소문자는 구분)



## ✓ table 생성 1

- 회원의 정보를 저장할 수 있는 ssafy\_member라는 이름의 table을 생성해보자.
  - 스키마 : 데이터베이스의 테이블에 저장될 데이터의 구조와 형식을 정의.

회원 정보 테이블			
컬럼명	타입	제약조건	설명
idx	int	auto_increment, PK	자동 증가 값
userid	varchar(16)	NOT NULL	사용자 아이디
username	varchar(20)		사용자 이름
userpwd	varchar(16)		사용자 비밀번호
emailid	varchar(20)		이메일 아이디
emaildomain	varchar(50)		이메일 도메인
joindate	timestamp	current_timestamp	가입일

### ✓ table 생성 2

- ER Diagram(ERD)
  - 개체 타입과 관계 타입을 기본 개념으로 현실 세계를 개념적으로 표현하는 방법.

회원정보	
🔑 일련번호	N/A INT
● 사용자아이디	N/A VARCHAR(16)
● 사용자이름	N/A VARCHAR(20)
● 사용자비밀번호	N/A VARCHAR(16)
● 이메일아이디	N/A VARCHAR(20)
● 이메일도메인	N/A VARCHAR(50)
● 가입일	N/A TIMESTAMP

logical diagram

ssafy_member	
🔑 idx	N/A INT
● userid	N/A VARCHAR(16)
● username	N/A VARCHAR(20)
● userpwd	N/A VARCHAR(16)
● emailid	N/A VARCHAR(20)
● emaildomain	N/A VARCHAR(50)
● joindate	N/A TIMESTAMP

physical diagram



### ✓ table 생성 3

- 회원의 정보를 저장할 수 있는 ssafy\_member라는 이름의 table을 생성해보자.
  - 스키마를 참조하여 테이블 생성 SQL 작성.

```
use ssafydb;

CREATE TABLE ssafy_member (
  idx          INT          NOT NULL AUTO_INCREMENT,
  userid       VARCHAR(16)  NOT NULL,
  username     VARCHAR(20),
  userpwd      VARCHAR(16),
  emailid      VARCHAR(20),
  emaildomain  VARCHAR(50),
  joindate     TIMESTAMP    NOT NULL DEFAULT current_timestamp,
  PRIMARY KEY (idx)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

## DML (Insert, Update, Delete)

## ✓ Data Manipulation Language (DML)

Confidential

SQL문	설명
insert (C)	데이터베이스 객체에 데이터를 입력.
select (R)	데이터베이스 객체에서 데이터를 조회.
update (U)	데이터베이스 객체에 데이터를 수정.
delete (D)	데이터베이스 객체에 데이터를 삭제.

## ✓ Data Manipulation Language (DML) : INSERT

- INSERT

- > INSERT INTO table\_name

- VALUES (col\_val1, col\_val2, col\_val3, ..., col\_valN);

- > INSERT INTO table\_name (col\_name1, col\_name2, col\_name3, ..., col\_nameN)

- VALUES (col\_val1, col\_val2, col\_val3, ..., col\_valN);

- > INSERT INTO table\_name (col\_name1, col\_name2, col\_name3, ..., col\_nameN)

- VALUES (col\_val1, col\_val2, col\_val3, ..., col\_valN),  
(col\_val1, col\_val2, col\_val3, ..., col\_valN);

- 생략이 가능한 field

1. NULL이 허용된 컬럼.
2. DEFAULT가 설정된 컬럼.
3. AUTO INCREMENT가 설정된 컬럼.

### ✓ Data Manipulation Language (DML) : INSERT

- ssafy\_member table에 정보 넣기.

- insert data :

사용자아이디: kimssafy,

이름: 김싸피,

비밀번호: 1234,

이메일아이디: kimssafy,

이메일 도메인: ssafy.com,

가입일: 오늘날짜

```
INSERT INTO ssafy_member (userid, username, userpwd, emailid, emaildomain, joindate)
VALUES ('kimssafy', '김싸피', '1234', 'kimssafy', 'ssafy.com', now());
```

```
INSERT INTO ssafy_member (username, userid , userpwd, joindate)
VALUES ('김싸피', 'kimssafy', '1234', now());
```

```
INSERT INTO ssafy_member (username, userid , userpwd, joindate)
VALUES
    ('김싸피', 'kimssafy', '1234', now()),
    ('박싸피', 'parkssafy', '9876', now());
```

## ✓ Data Manipulation Language (DML) : UPDATE

- UPDATE
  - > UPDATE table\_name  
SET col\_name1 = col\_val1, [ col\_name2 = col\_val2, ..., col\_nameN = col\_valN]  
WHERE conditions;
- WHERE절의 conditions(조건)에 만족하는 레코드의 값을 변경.
- 주의 : WHERE절을 생략하면 모든 데이터가 바뀐다.

### ✓ Data Manipulation Language (DML) : UPDATE

- ssafy\_member table에 정보 변경.

- 변경 data :

사용자아이디: [kimssafy](#)인 회원의 정보를 다음으로 변경.

비밀번호: [9876](#) , 이메일 도메인: [ssafy.co.kr](#)

```
UPDATE ssafy_member  
SET userpwd = 9876, emaildomain = 'ssafy.co.kr'  
WHERE userid = 'kimssafy';
```

## ✓ Data Manipulation Language (DML) : DELETE

- DELETE
  - > DELETE form table\_name  
WHERE conditions;
- WHERE절의 conditions(조건)에 만족하는 레코드의 값을 삭제.
- 주의 : WHERE절을 생략하면 모든 데이터가 삭제된다.



## ✓ Data Manipulation Language (DML) : DELETE

- ssafy\_member table에 정보 삭제.

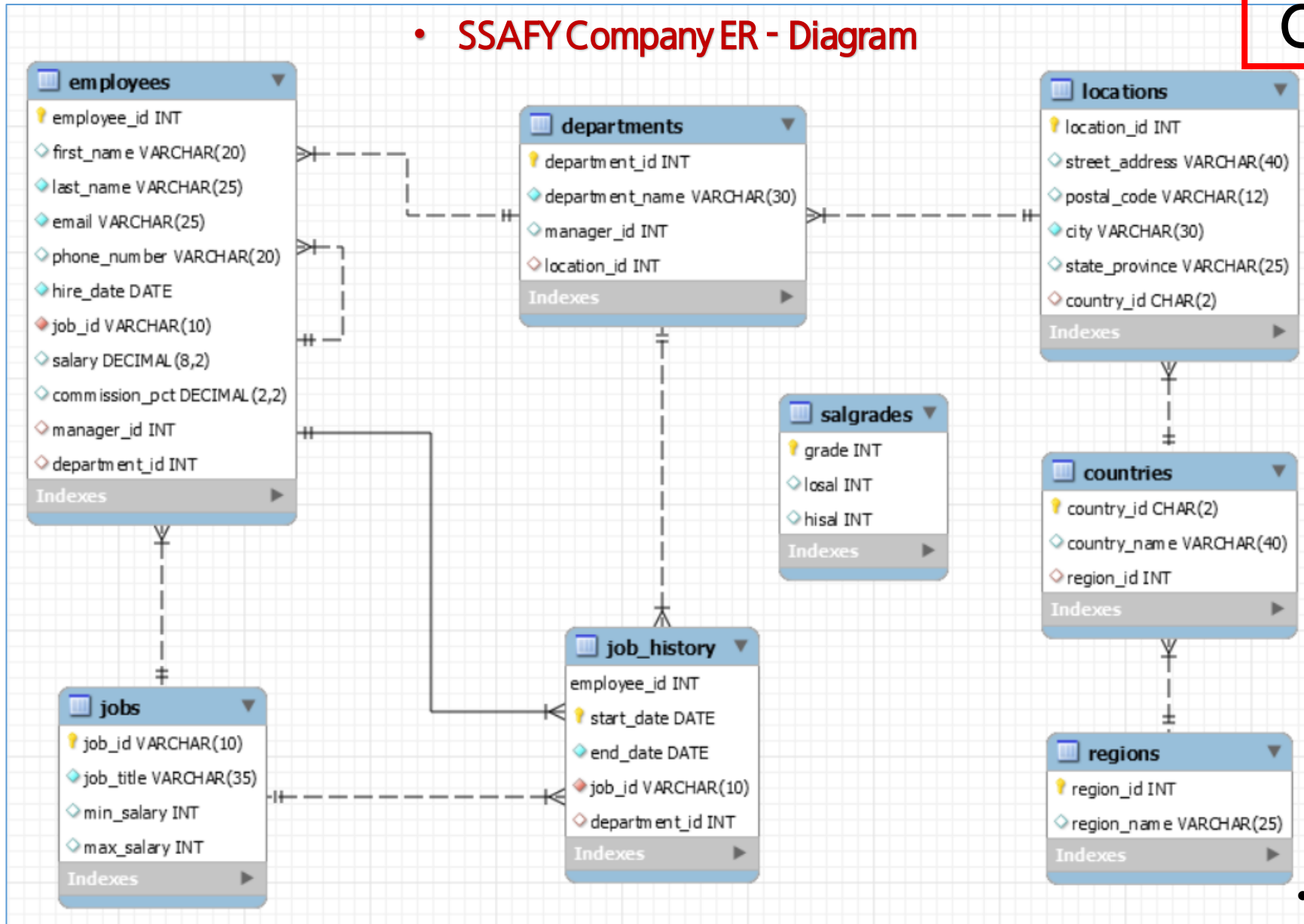
- 삭제 data :

사용자아이디 : kimssafy인 회원의 정보를 삭제.

```
DELETE from ssafy_member  
WHERE userid = 'kimssafy';
```

## DML (Select)

- SSAFY Company ER - Diagram



## ✓ Data Manipulation Language (DML) : SELECT

Confidential

- SELECT
  - `SELECT * | { [ ALL | DISTINCT ] column | expression [ alias ], ... }`  
`FROM table_name;`
- SELECT clause와 FROM clause은 필수.

select clause	description
*	FROM 절에 나열된 테이블에서 모든 열을 선택.
ALL	선택된 모든 행을 반환. ALL이 default (생략 가능).
DISTINCT	선택된 모든 행 중에서 중복 행 제거.
column	FROM 절에 나열된 테이블에서 지정된 열을 선택.
expression	표현식은 값으로 인식되는 하나 이상의 값, 연산자 및 SQL 함수의 조합을 뜻함.
alias	별칭.

### ✓ Data Manipulation Language (DML) : SELECT

- 기본 SELECT.

- 모든 사원의 모든 정보 검색.

```
select *  
from employees;
```

- 사원이 근무하는 부서의 부서번호 검색.

```
SELECT all department_id  
FROM employees;
```

```
select distinct employee_id  
from employees;
```

\* 회사에 존재하는 모든 부서.

```
select department_id  
from departments;
```

- 모든 사원의 사번, 이름, 급여 검색.

```
select employee_id, first_name, salary  
from employees;
```

### ✓ Data Manipulation Language (DML) : SELECT

- alias, 사칙연산 (+, -, \*, /), NULL Value.
  - 모든 사원의 사번, 이름, 급여, 급여 \* 12 (연봉) 검색.

```
select employee_id as 사번, first_name "이름",  
       salary as "급여", salary * 12 "연봉"  
from employees;
```

- 모든 사원의 사번, 이름, 급여, 급여 \* 12 (연봉), 커미션, 커미션포함 연봉 검색.

```
select employee_id 사번, first_name "이름", salary "급여",  
       salary * 12 "연봉", commission_pct,  
       (salary + salary * commission_pct) * 12 "커미션포함연봉1",  
       (salary + salary * IFNULL(commission_pct, 0)) * 12 "커미션포함연봉2"  
from employees;
```

IFNULL(expr1, expr2) : expr1이 null이면 expr2가 return.

### ✓ Data Manipulation Language (DML) : SELECT

- CASE exp1 WHEN exp2 THEN exp3  
[ WHEN exp4 THEN exp5  
...  
ELSE exp6 ]

END

- 모든 사원의 사번, 이름, 급여, 급여에 따른 등급표시 검색.

15000 이상 “고액연봉”    8000 이상 “평균연봉”    8000 미만 “저액연봉 ”

```
select employee_id, first_name, salary,  
       case when salary > 15000 then '고액연봉'  
            when salary > 8000 then '평균연봉'  
            else '저액연봉'  
       end "연봉등급"  
from employees;
```

## ✓ Data Manipulation Language (DML) : SELECT

Confidential

- SELECT
  - `SELECT * | { [ ALL | DISTINCT ] column | expression [ alias ], ... }`  
`FROM table_name`  
`WHERE conditions;`
- WHERE clause : 조건에 만족하는 행을 검색.



### ✓ Data Manipulation Language (DML) : SELECT

- AND, OR, NOT.

- 부서번호가 50인 직원중 급여가 7000이상인 직원의 사번, 이름, 급여, 부서번호 검색.

```
select employee_id, first_name, salary, department_id
from employees
where department_id = 50
and salary >= 7000;
```

- 근무 부서번호가 50, 60, 70에 근무하는 직원의 사번, 이름, 부서번호 검색.

```
select employee_id, first_name, department_id
from employees
where department_id = 50
or department_id = 60
or department_id = 70;
```

## ✓ Data Manipulation Language (DML) : SELECT

- AND, OR, NOT.

- 근무 부서번호가 50, 60, 70이 아닌 사원의 사번, 이름, 부서번호 검색.

```
select employee_id, first_name, department_id
from employees
where department_id != 50
and department_id != 60
and department_id != 70;
```

```
select employee_id, first_name, department_id
from employees
where not (department_id = 50
or department_id = 60
or department_id = 70);
```

## ✓ Data Manipulation Language (DML) : SELECT

- IN.

- 근무 부서번호가 50, 60, 70에 근무하는 사원의 사번, 이름, 부서번호 검색.

```
select employee_id, first_name, department_id
from employees
where department_id in (50, 60, 70);
```

- 근무 부서번호가 50, 60, 70이 아닌 사원의 사번, 이름, 부서번호 검색.

```
select employee_id, first_name, department_id
from employees
where department_id not in (50, 60, 70);
```

## ✓ Data Manipulation Language (DML) : SELECT

- BETWEEN.
  - 급여가 6000이상 10000이하인 사원의 사번, 이름, 급여 검색.

```
select employee_id, first_name, salary
from employees
where salary >= 6000
and salary <= 10000;
```


```
select employee_id, first_name, salary
from employees
where salary between 6000 and 10000;
```

### ✓ Data Manipulation Language (DML) : SELECT

- NULL 비교 : IS NULL, IS NOT NULL.

- 근무 부서가 지정되지 않은(알 수 없는) 사원의 사번, 이름, 부서번호 검색.

```
select employee_id, first_name, salary
from employees
where department_id = null;
```



```
select employee_id, first_name, salary
from employees
where department_id is null;
```

- 커미션을 받는 사원의 사번, 이름, 급여, 커미션 검색.

```
select employee_id, first_name, salary, commission_pct
from employees
where commission_pct is not null;
```

### ✓ Data Manipulation Language (DML) : SELECT

- LIKE (wild card : %, \_).
  - 이름에 'x'가 들어간 사원의 사번, 이름 검색.

```
select employee_id, first_name
from employees
where first_name like '%x%';
```

- 이름의 끝에서 3번째 자리에 'x'가 들어간 사원의 사번, 이름 검색.

```
select employee_id, first_name
from employees
where first_name like '%x__';
```

### ✓ Data Manipulation Language (DML) : SELECT

- 논리연산시 주의점 : NULL.

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

## ✓ Data Manipulation Language (DML) : SELECT

Confidential

- SELECT
  - `SELECT * | { [ ALL | DISTINCT ] column | expression [ alias ], ... }`  
`FROM table_name`  
`WHERE conditions`  
`ORDER BY col_name1 [ ASC | DESC ] [ , col_name2, ... ];`
- ORDER BY clause : 정렬 (default : ASC)



## ✓ Data Manipulation Language (DML) : SELECT

- 정렬.

- 모든 사원의 사번, 이름, 급여 검색  
단, 급여 순 정렬(내림차순)

```
select employee_id, first_name, salary
from employees
order by salary desc;
```

- 50, 60, 70에 근무하는 사원의 사번, 이름, 부서번호, 급여 검색.  
단, 부서별 정렬(오름차순) 후 급여 순(내림차순) 검색

```
select employee_id, first_name, department_id, salary
from employees
order by department_id, salary desc;
```

# JDBC(Java DataBase Connectivity)

## ✓ JDBC(Java DataBase Connectivity)란?

- 자바 프로그래밍 언어로 만들어진 클래스와 인터페이스로 이루어진 API로서 ANSI SQL(1999)를 지원.
- SQL문을 실행할 수 있는 함수 호출 인터페이스이다.

## ✓ JDBC 특징.

- DBMS 종류에 독립적인 자바 프로그래밍 가능.
- 데이터베이스가 달라지더라도 동일한 API를 사용하게 해준다(드라이버 및 URL만 수정 하면 가능).
- 자바가 가지는 플랫폼에 독립적이라는 특성과 DBMS에 독립적인 특성을 가진다.

## ✓ JDBC 기능.

- 데이터베이스에 연결 설정 한다
- SQL문장을 DBMS에 전송한다
- SQL문장 전송 후 결과를 처리할 수 있게 해준다.

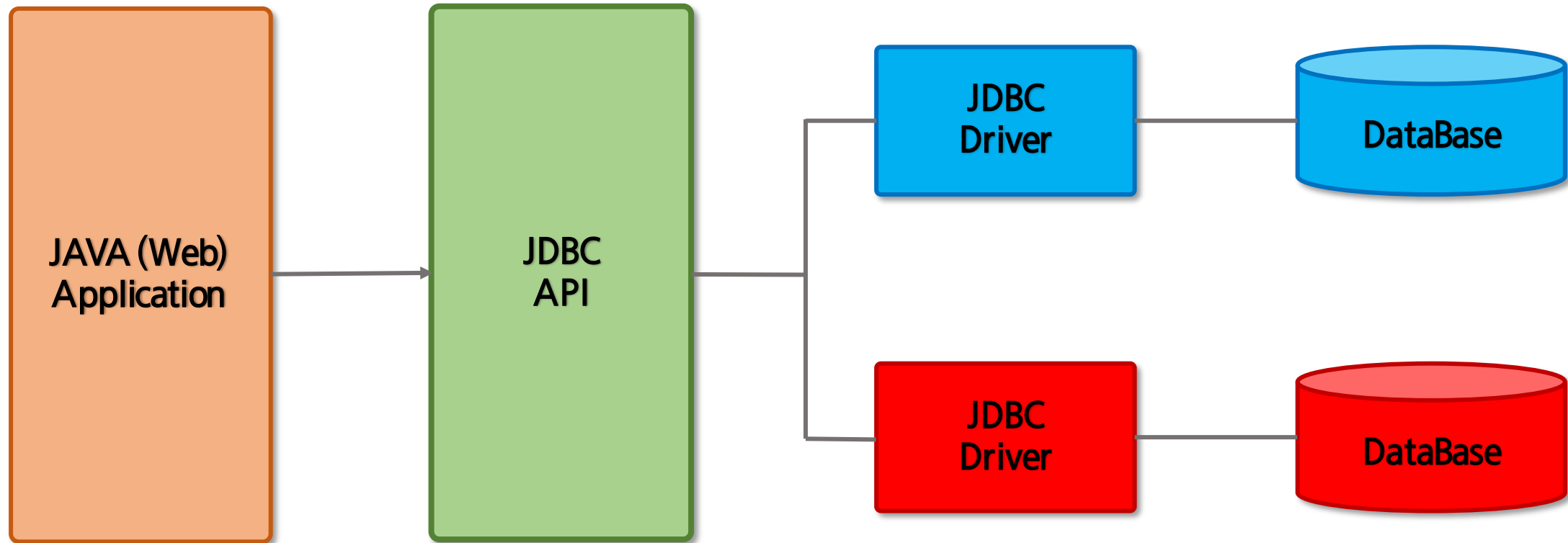
## ✓ JDBC Interface.

- Database를 만드는 업체에게 제공되는 인터페이스
  - 업체에게 제공되는 인터페이스를 각각의 DBMS업체들이 구현해 놓은 것으로서 이것이 바로 드라이버이다.
- 프로그래머에게 제공되는 인터페이스
  - SQL 패키지가 제공하고 있는 라이브러리로서 프로그래머는 이 라이브러리를 기반으로 DB 프로그램을 작성할 수 있다.

# JDBC (Java DataBase Connectivity)

✓ JDBC.

Confidential



✓ JDBC API : java.sql package

- Driver (interface)

- 드라이버에 대한 정보를 가지고 있다.
- 모든 드라이버가 반드시 구현해야 하는 인터페이스.
- 드라이버의 버전이나 연결에 대한 정보를 알아볼 수 있는 메소드를 가지고 있다.

[Java Platform, Standard Edition 8 API Specification](#)

✓ JDBC API : java.sql package

- Connection (interface)

- 데이터베이스에 대한 하나의 세션을 표현한다.
  - ✓ 세션은 하나의 클라이언트가 서버에 요청을 하기 위해 연결을 맺은 상태를 의미한다.
- DriverManager 클래스의 getConnection() 메소드를 이용하여 얻어 올 수 있다.
- 디폴트로 setAutoCommit(true)로 설정된다.
- 개발자가 원하는 경우에 commit을 해주고 싶거나 트랜잭션이 아주 중요한 부분에 있어서 RollBack 처리를 하고자 할 경우에는 setAutoCommit(false)로 설정한다.
  - ✓ 단, 이 경우에는 SQL문을 수행할 때 마다 명시적으로 commit()을 호출해야 한다.

✓ JDBC API : java.sql package

- Statement (interface)

- SQL문장을 실행하고 그것에 대한 결과 값을 가져오기 위해 사용한다.
- public boolean execute(String sql) throws SQLException
  - ✓ 특별히 SQL문을 구분하지 않고 DML(delete, update, insert), Query(select), DDL(create, drop) 등을 수행할 수 있다. 결과가 ResultSet이면 true이고 결과가 DML이거나 특별한 결과가 없으면 false를 리턴한다.
- public ResultSet executeQuery(String sql) throws SQLException
  - ✓ Select를 처리할 때 사용한다.
- public int executeUpdate(String sql) throws SQLException
  - ✓ 주로 DML(delete, update, insert)등의 SQL을 수행할 때 사용한다.



✓ JDBC API : java.sql package

- PreparedStatement (interface)

- 동일한 SQL 문장이 여러 번 반복적으로 수행될 때 사용하는 객체.
- 대용량의 문자나 바이너리 타입의 데이터(이미지나 사운드 등)를 저장하기 위해서도 사용될 수 있다.
- SQL 문장이 미리 컴파일 되어 PreparedStatement 객체에 저장된다.
- 여러 번 반복 수행 시 clearParameters() 메소드를 이용해 Statement에 남겨진 값을 초기화 한다.
- public ResultSet executeQuery() throws SQLException
  - ✓ Select를 처리할 때 사용한다.
- public int executeUpdate() throws SQLException
  - ✓ 주로 DML(delete, update, insert)등의 SQL을 수행할 때 사용한다.

✓ JDBC API : java.sql package

- CallableStatement (interface)

- 데이터베이스에 대하여 실제 SQL문을 실행하는 것이 아니라 Stored Procedures를 호출한다.
- Stored Procedures란 연속되는 SQL문으로 데이터베이스에 저장해 두고 마치 함수의 호출처럼 사용한다.
- 데이터베이스에 Stored Procedures를 만들어 두고 자바에서 호출하여 사용할 수 있게 한다.
- Stored Procedures 사용 시 속도의 향상을 기대할 수 있고, 자바 코드에 SQL문장이 들어가지 않으므로 자바 코드가 SQL에 독립적이 된다.

✓ JDBC API : java.sql package

- ResultSet (interface)

- 쿼리에 대한 결과값 처리.
- ResultSet 객체의 커서는 첫번째 레코드보다 바로 이전을 가르킨다.
- next()
  - ✓ ResultSet 객체의 커서를 이동
- getXXX(index or name) 메소드를 이용하여 데이터를 얻을 수 있다.
  - ✓ getString(index or name);
  - ✓ getInt(index or name);
  - ✓ getDate(index or name);
  - ✓ ...

## ✓ JDBC Programming 개발 순서.

1. JDBC Driver Loading.
2. DBMS와 연결. (Connection 생성)
3. SQL 실행 준비. (Statement, PreparedStatement 생성)
4. SQL 실행.
  - DML (insert, update, delete)
  - Query (select)
5. DBMS 연결 끊기.

## ✓ JDBC Programming 개발 순서.

### 1. JDBC Driver Loading.

- 데이터베이스와의 접속을 오픈 하기 위해 애플리케이션의 JVM안으로 특정 드라이버 클래스를 적재.
- `Class.forName( Driver ClassName );`
- 각 DataBase별 Driver Class
  - ✓ MySQL : `com.mysql.cj.jdbc.Driver`
  - ✓ Oracle : `oracle.jdbc.driver.OracleDriver`
  - ✓ MSSQL : `com.microsoft.sqlserver.jdbc.SQLServerDriver`

## ✓ JDBC Programming 개발 순서.

### 2. DBMS와 연결. (Connection 생성)

- DriverManager 클래스를 이용하여 URL 형태로 주어진 데이터 베이스에 대한 접속을 요청.
- `Connection con = DriverManager.getConnection(URL, dbid, dbpassword);`
- JDBC URL은 드라이버 고유의 방식으로 개별적인 데이터베이스를 식별.
  - ✓ MySQL : `jdbc:mysql://HOST:PORT/DBNAME[?param1=value1&param2=value2&..]`
  - ✓ Oracle : `jdbc:oracle:thin:@HOST:PORT:SID`
  - ✓ MSSQL : `jdbc:sqlserver://HOST:PORT;databaseName=DB`

## ✓ JDBC Programming 개발 순서.

### 3. SQL 실행 준비. (Statement, PreparedStatement 생성)

- String sql = "insert, update, delete, select, ...";
- Statement 생성.
  - ✓ Statement stmt = conn.createStatement();
- PreparedStatement 생성.
  - ✓ PreparedStatement pstmt = conn.prepareStatement(sql);

## ✓ JDBC Programming 개발 순서.

### 4. SQL 실행. (executeUpdate, executeQuery)

- Statement.
  - ✓ DML
    - `int cnt = stmt.executeUpdate(sql);`
  - ✓ Select
    - `ResultSet rs = stmt.executeQuery(sql);`
- PreparedStatement.
  - ✓ sql문의 치환변수 값 설정 : `pstmt.setXXX(index, val);`
  - ✓ DML
    - `int cnt = pstmt.executeUpdate();`
  - ✓ Select
    - `ResultSet rs = pstmt.executeQuery();`



## ✓ JDBC Programming 개발 순서.

### 4. SQL 실행. (executeUpdate, executeQuery)

- ResultSet.
  - ✓ ResultSet을 얻어온 후 rs.next()를 실행해야 한다.
    - select의 결과가 반드시 하나가 나오는 경우. : rs.next();
    - select의 결과가 하나 또는 못 얻어 오는 경우. : if(rs.next) { }
    - select의 결과가 여러 개가 나올 수 있는 경우. : while(rs.next()) { }
  - ✓ 값 얻기.
    - String str = rs.getString(index or name);
    - int cnt = rs.getInt(index or name);

## ✓ JDBC Programming 개발 순서.

### 5. DBMS 연결 끊기.

- 모든 작업이 끝난 경우에는 ResultSet, Statement(PreparedStatement), Connection 객체의 close() 메소드를 이용하여 작업을 종료한다. (연결한 순의 역순으로 종료)
- Connection은 상당한 Overhead를 가져온다. 따라서 최적화된 상태를 유지하기 위해서는 **반드시** Connection을 닫아 주어야 한다.
  - ✓ rs.close();
  - ✓ pstmt.close();
  - ✓ conn.close();

# 내일 방송에서 만나요!

삼성 청년 SW 아카데미