

# 삼성 청년 SW 아카데미

MySQL

## <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.

# 목차

---

1. RDBMS & SQL
2. DML (Insert, Update, Delete)
3. DML (Select)

# RDBMS & SQL

삼성 청년 SW 아카데미

## ✓ RDBMS ?

- 관계형(Relational) 데이터베이스 시스템.
- 테이블기반(Table based)의 DBMS.
  - 데이터를 테이블 단위로 관리.
    - >> 하나의 테이블은 여러 개의 컬럼(Column)으로 구성.
  - 중복 데이터를 최소화 시킴.
    - >> 같은 데이터가 여러 컬럼 또는 테이블에 존재 했을 경우,  
데이터를 수정 시 문제가 발생할 가능성이 높아짐. - 정규화.
  - 여러 테이블에 분산되어 있는 데이터를 검색 시 테이블 간의 관계(join)를 이용하여 필요한 데이터를 검색.

## ✓ RDBMS의 저장 구조.

- Table

열(Column)

Column Name

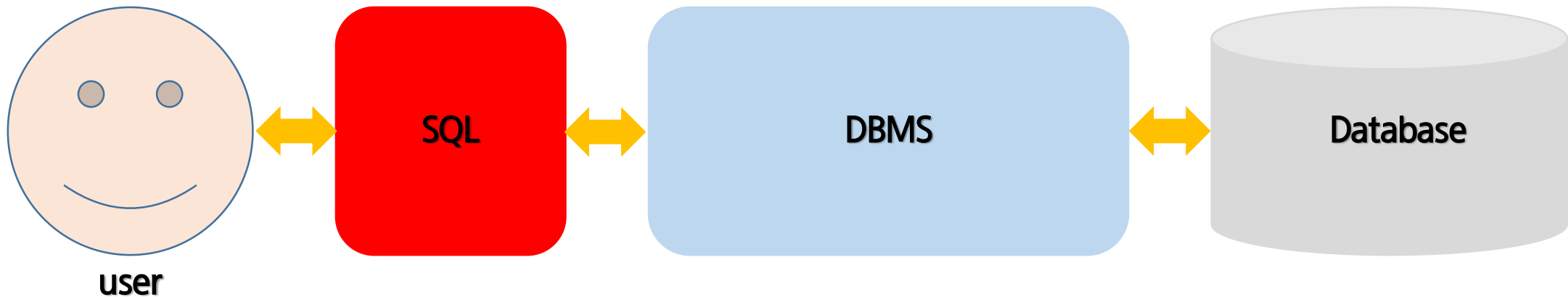
employee_id	first_name	last_name	email	phone_number	hire_date	job_id	salary	commission_pct	manager_id	department_id
100	Steven	King	SKING	515.123.4567	1987-10-17	AD_PRES	24000.00	NULL	NULL	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	1989-11-21	AD_VP	17000.00	NULL	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	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)

## ✓ SQL (Structured Query Language)

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



# SQL (Structured Query Language)

✓ SQL 구문은 DCL, DDL, DML로 구분하며, 아래와 같은 종류가 있다.

문장	설명
INSERT UPDATE DELETE	DML(Data Manipulation Language)이라 부르며, 개별적으로 Database 테이블에서 새로운 행을 입력하고, 기존의 행을 변경하고 제거한다.
SELECT	Database로 부터 Data를 검색합니다. SELECT 역시 DML로 분류된다.
CREATE ALTER DROP RENAME	DDL(Data Definition Language)이라 부르며, 테이블로부터 데이터 구조를 생성, 변경, 제거한다.
COMMIT ROLLBACK	DML 명령문으로 수행한 변경을 관리한다.
GRANT REVOKE	DCL(Data Control Language)이라 부르며, Database와 그 구조에 대한 접근권한을 제공하거나 제거한다.



# DDL (Data Definition Language)

## ✓ SQL 종류.

- DDL (Data Definition Language) : 데이터 정의어.
  - 데이터베이스 객체(table, view, index,...)의 구조를 정의.
  - 테이블 생성, 컬럼 추가, 타입변경, 제약조건 지정, 수정등.

SQL문		설명
create	데이터베이스 객체를 생성.	
drop	데이터베이스 객체를 삭제.	
alter	기존에 존재하는 데이터베이스 객체를 수정.	

- live\_DML.sql

# DML (Data Manipulation Language)

## ✓ SQL 종류.

- DML (Data Manipulation Language) : 데이터 조작어.
  - Data 조작기능.
  - 테이블의 레코드를 CRUD (Create, Retrieve, Update, Delete)

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.	

# DCL (Data Control Language)

## ✓ SQL 종류.

- DCL (Data Control Language) : 데이터 제어어.
  - DB, Table의 접근권한이나 CRUD 권한을 정의.
  - 특정 사용자에게 테이블의 검색권한 부여/금지등.

SQL문		설명
grant	데이터베이스 객체에 권한을 부여.	
revoke	데이터베이스 객체 권한 취소.	

# TCL (Transaction Control Language)

## ✓ SQL 종류.

- TCL (Transaction Control Language) : 트랜잭션 제어어.
  - transaction란 데이터베이스의 논리적 연산 단위.

SQL문	설명
commit	실행한 Query를 최종적으로 적용.
rollback	실행한 Query를 마지막 commit 전으로 취소시켜 데이터를 복구.

## ✓ 데이터베이스 생성.

- 데이터베이스 생성.

> **create database** 데이터베이스명;

> **create database** 데이터베이스명

**default character set** 값

**collate** 값;

> Character set은 각 문자가 컴퓨터에 저장될 때 어떠한 '코드'로 저장 될지에 대한 규칙의 집합을 의미한다.

> Collation은 특정 문자 셋에 의해 데이터베이스에 저장된 값들을 비교 검색하거나 정렬 등의 작업을 위해 문자들을 서로 '비교' 할 때 사용하는 규칙들의 집합을 의미한다.

- 다국어 처리(utf8mb3) : dbtest 생성.

```
create database dbtest
default character set utf8mb3 collate utf8mb3_general_ci;
```

- 이모지 문자까지 처리.

```
create database dbtest
default character set utf8mb4 collate utf8mb4_general_ci;
```

## ✓ 데이터베이스 변경.

- 데이터베이스 변경.

> **alter database** 데이터베이스명  
**default character set** 값 **collate** 값;

- dbtest의 character set, collate 변경.

```
alter database dbtest  
default character set utf8mb4 collate utf8mb4_general_ci;
```

## ✓ 데이터베이스 삭제.

- 데이터베이스 삭제.

> **drop database** 데이터베이스명;

- 이름이 'dbtest'인 데이터베이스 삭제.

```
drop database dbtest;
```

- 데이터베이스 사용.

> **use** 데이터베이스명;

- 이름이 ssafydb인 데이터베이스 사용.

```
use ssafydb;
```

# Data Type - 문자형

## ✓ table 생성.

- Data Type
  - 문자형 데이터 타입

데이터 유형	정의
CHAR[(M)]	고정 길이를 갖는 문자열을 저장. M은 1 ~ 255( $2^8 - 1$ ). CHAR(20)인 컬럼에 10자만 저장을 하더라도, 20자 만큼의 기억장소를 차지.
VARCHAR[(M)]	가변 길이를 갖는 문자열을 저장. M은 1 ~ 65535( $2^{16} - 1$ ). VARCHAR(20)인 컬럼에 10자만 저장을 하면, 실제로도 10자 만큼의 기억장소를 차지.
TINYTEXT[(M)]	최대 255( $2^8 - 1$ )byte
TEXT[(M)]	최대 65535( $2^{16} - 1$ )byte
MEDIUMTEXT[(M)]	최대 16777215( $2^{24} - 1$ )byte
LONGTEXT[(M)]	최대 4294967295( $2^{32} - 1$ )byte
ENUM('value1','value2',...)	열거형. 정해진 몇가지의 값들 중 하나만 저장. 최대 65535개의 개별 값을 가질 수 있고, 내부적으로 정수 값으로 표현된다.
SET('value1','value2',...)	집합형. 정해진 몇가지의 값들 중 여러 개를 저장. 최대 64개의 요소로 구성될 수 있고, 내부적으로는 정수 값이다.



# Data Type - 숫자형

## ✓ table 생성.

- Data Type
  - 숫자형 데이터 타입1

데이터 유형	바이트	정의
BIT[(M)]	1	비트 값 유형. M은 값 당 비트 수를 나타내며 1에서 64 사이의 값을 나타냄.
BOOL, BOOLEAN		이 유형은 TINYINT (1)의 동의어. 0은 false, 0이 아닌 값은 true로 간주
TINYINT[(M)]	1	(signed) -128 ~ 127 (unsigned) 0 ~ 255( $2^8$ )
SMALLINT[(M)]	2	(signed) -32768 ~ 32767 (unsigned) 0 ~ 65535( $2^{16}$ )
MEDIUMINT[(M)]	3	(signed) -8388608 ~ 8388607 (unsigned) 0 ~ 16777215( $2^{24}$ )
INT[(M)]	4	(signed) -2147483648 ~ 2147483647 (unsigned) 0 ~ 4294967295( $2^{32}$ )
BIGINT[(M)]	8	(signed) -9223372036854775808 ~ 9223372036854775807 (unsigned) 0 ~ 18446744073709551615( $2^{64}$ )
FLOAT[(M, D)]	4	(signed) -3.402823466E+38 ~ 1.175494351E-38 (unsigned) 1.175494351E-38 ~ 3.402823466E+38

# Data Type - 숫자형

## ✓ table 생성.

- Data Type
  - 숫자형 데이터 타입2

데이터 유형	바이트	정의
DOUBLE[(M, D)] DOUBLE PRECISION[(M, D)] REAL[(M, D)]	8	(signed) -1.7976931348623157E+908 ~ -2.2250738585072014E-308 (unsigned) 2.2250738585072014E-308 ~ 1.7976931348623157E+308
FLOAT(p)		부동 소수점 숫자. p는 비트 정밀도를 가리키지만, MySQL은 결과 데이터 타입으로 FLOAT 또는 DOUBLE을 사용할 지를 결정할 때에만 이 값을 사용한다.
DECIMAL[(M[, D])]	길이+1	묶음 고정 소수점 숫자 M은 전체 자릿수(Precision : 정밀도), D는 소수점 뒤 자릿수(Scale : 배율)  - DECIMAL(5)의 경우 : -99999 ~ 99999 - DECIMAL(5, 1)의 경우 : -9999.9 ~ 9999.9 - DECIMAL(5, 2)의 경우 : -999.99 ~ 999.99  최대 65자리까지 지원
DEC[(M[, D])] NUMERIC[(M[, D])] FIXED[(M[, D])]		DECIMAL과 동의어다. FIXED 동의어는 다른 데이터베이스 시스템과의 호환을 위해서 사용하는 것이다.

# Data Type - 날짜형

## ✓ table 생성.

- Data Type
  - 날짜형 데이터 타입

데이터 유형	바이트	정의
DATE	3	YYYY-MM-DD('1001-01-01' ~ '9999-12-31')
TIME	3	HH:MM:SS('-838:59:59' ~ '838:59:59')
DATETIME	8	YYYY-MM-DD HH:MM:SS('1001-01-01 00:00:00' ~ '9999-12-31 23:59:59')
TIMESTAMP[(M)]	4	1970-01-01 ~ 2038-01-19 03:14:07까지 지원(1970-01-01 00:00:00 를 0으로 해서 1초단위로 표기) Index가 더 빠르게 생성.
YEAR[(2 4)]	1	2와 4를 지정할 수 있으며, 2인 경우에 값의 범위는 70 ~ 69, 4인 경우에는 1970 ~ 2069이다

# Data Type - 이진 데이터형

## ✓ table 생성.

- Data Type
  - 이진 데이터 타입

데이터 유형	정의
BINARY[(M)]	CHAR 유형과 유사하지만 이진 바이트 문자열을 이진이 아닌 문자열로 저장. M은 바이트 단위의 열 길이를 나타냄.
VARBINARY[(M)]	VARCHAR 유형과 유사하지만 이진 바이트 문자열을 이진이 아닌 문자열로 저장. M은 바이트 단위의 열 길이를 나타냄.
TINYBLOB[(M)]	이진 데이터 타입. 최대 $255(2^8 - 1)$ byte
BLOB[(M)]	이진 데이터 타입. 최대 $65535(2^{16} - 1)$ byte
MEDIUMBLOB[(M)]	이진 데이터 타입. 최대 $16777215(2^{24} - 1)$ byte
LOB[(M)]	이진 데이터 타입. 최대 $4294967295(2^{32} - 1)$ byte

## ✓ table 생성

- create table : <https://dev.mysql.com/doc/refman/8.0/en/create-table.html>

```
> create table table_name (  
    column_name1 Type [optional attributes],  
    column_name2 Type,  
    .  
    column_nameN Type,  
);
```

- optional attributes
  - NOT NULL : 각 행은 해당 열의 값을 포함해야 하며 null값은 허용되지 않음.
  - DEFAULT value : 값이 전달되지 않을 때 추가되는 기본값 설정.
  - UNSIGNED : Type이 숫자인 경우만 해당되며 숫자가 0 또는 양수로 제한됨.
  - AUTO INCREMENT : 새 레코드가 추가 될 때마다 필드 값을 자동으로 1증가시킴.
  - PRIMARY KEY : 테이블에서 행을 고유하게 식별하기 위해 사용. PRIMARY KEY 설정이 있는 열은 일반적으로 ID번호이며 AUTO INCREMENT와 같이 사용되는 경우가 많음.

## ✓ table 생성

- 제약 조건
  - 컬럼에 저장될 데이터의 조건을 설정하는 것.
  - 제약조건을 설정하면 조건에 위배되는 데이터는 저장 불가.
  - 테이블 생성시 컬럼에 직접 지정하거나 constraint로 지정, 또는 ALTER를 이용하여 설정가능.

제약 조건	description
NOT NULL	컬럼에 NULL값을 저장할 수 없고, 반드시 쿼리문을 이용하여 값을 지정.
UNIQUE	컬럼에 중복된 값을 저장 할 수 없음, NULL값은 허용.
PRIMARY KEY	컬럼에 중복된 값을 저장 할 수 없고, NULL값도 허용하지 않음. 주로 ROW를 구분하기 위한 유일한 값을 지정할 때 사용. '기본키'라고도 부름.
FOREIGN KEY	특정 테이블의 PK 컬럼에 저장되어 있는 값만 저장. '참조키', '외래키'라고도 부름. NULL값은 허용. references를 이용하여 어떤 컬럼에 어떤 데이터를 참조하는지 반드시 지정.
DEFAULT	NULL값이 들어올 경우 기본 설정되는 값을 지정.
CHECK	값의 범위나 종류를 지정. MYSQL 8.0.16부터 사용가능. (이전 버전의 경우 설정은 되나 적용이 안됨)

## ✓ 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;
```

# 이어서..

삼성 청년 SW 아카데미

# DML

(Insert, Update, Delete)

삼성 청년 SW 아카데미

## ✓ Data Manipulation Language (DML)

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

# DML - DELETE

## ✓ Data Manipulation Language (DML) : DELETE

- ssafy\_member table에 정보 삭제.

- 삭제 data :

사용자아이디 : kimssafy인 회원의 정보를 삭제.

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

# 이어서..

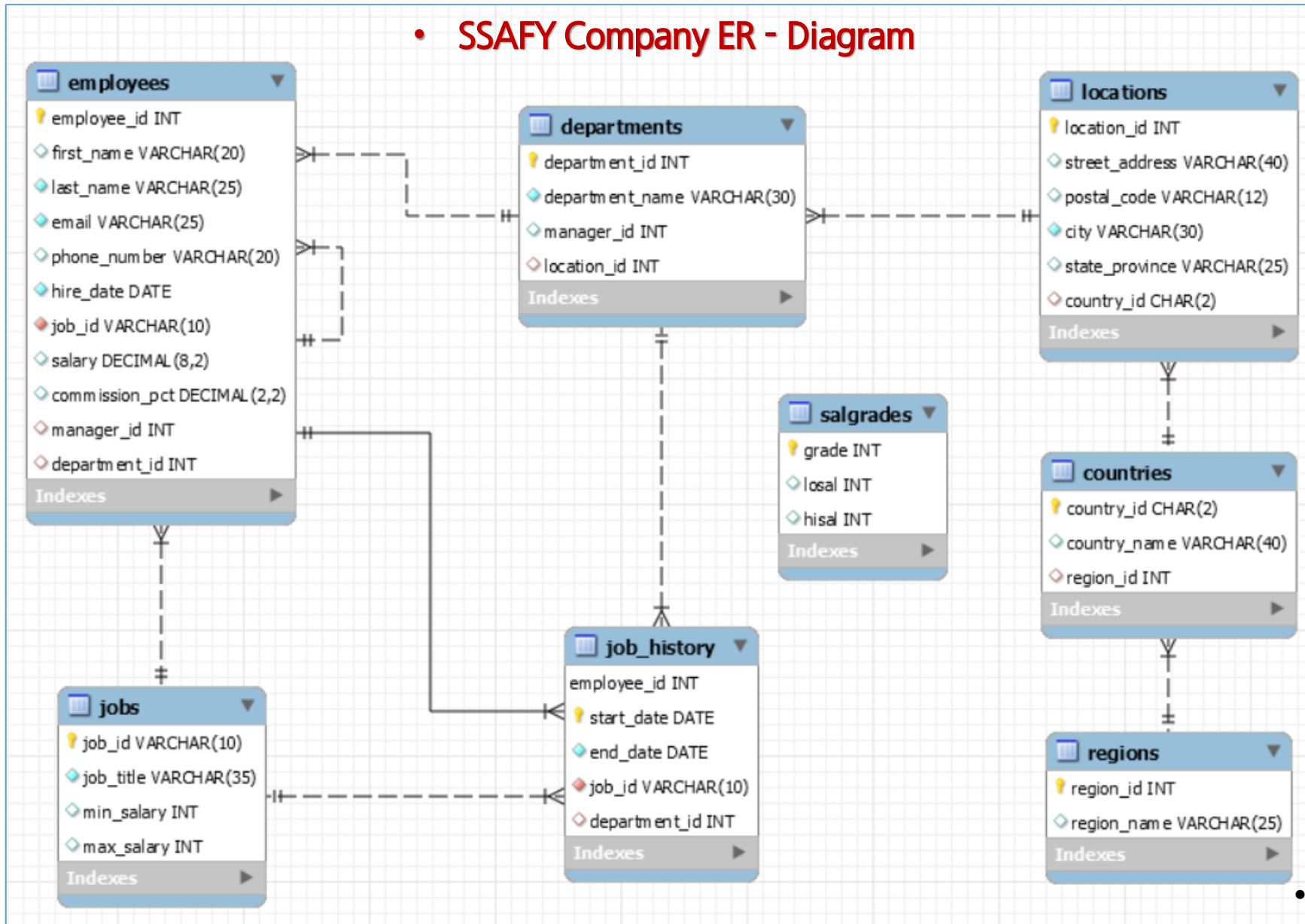
삼성 청년 SW 아카데미

# DML (Select)

삼성 청년 SW 아카데미

# DML - SELECT

- SSAFY Company ER - Diagram



## ✓ Data Manipulation Language (DML) : SELECT

- 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 "이 름",  
       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 "이 름", 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

- 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

- 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;
```

# 내일 방송에서 만나요!

삼성 청년 SW 아카데미