DBMS와 SQL

DBMS 개요

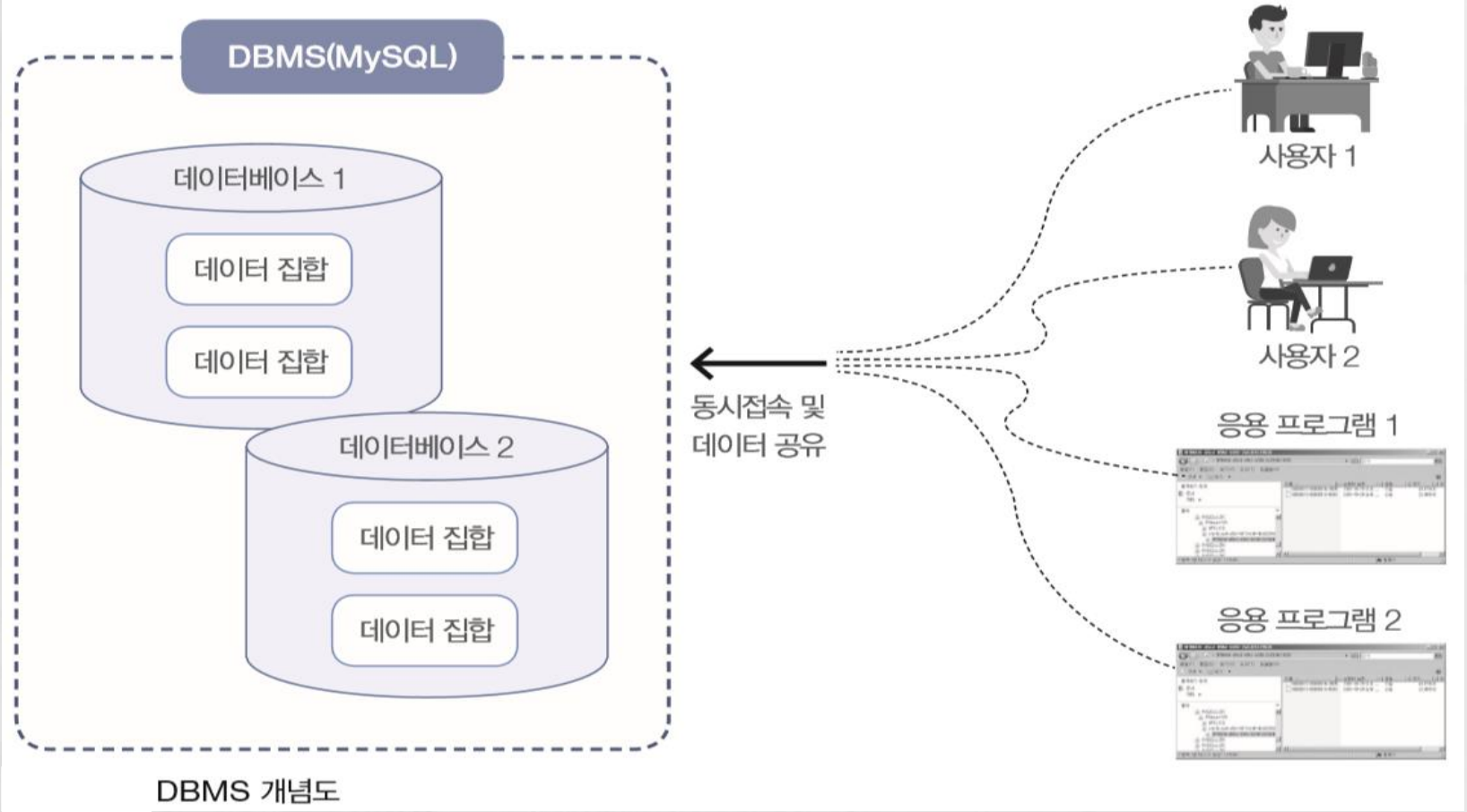
데이터베이스의 정의와 특징

- 데이터베이스
  - ‘데이터의 집합’
  - 여러 명의 사용자나 응용프로그램이 공유하는 데이터들
  - 동시에 접근 가능해야
  - 데이터의 저장 공간’ 자체
- DBMS
  - 데이터베이스를 관리·운영하는 역할

DBMS 개요

데이터베이스의 정의와 특징

- DBMS 개념도



DBMS 개념도

DBMS 개요

DB/DBMS의 특징

- 데이터의 무결성 (Integrity)
  - 데이터베이스 안의 데이터는 오류가 없어야
  - 제약 조건(Constrain)이라는 특성을 가짐
- 데이터의 독립성
  - 데이터베이스 크기 변경하거나 데이터 파일의 저장소 변경 시 기존에 작성된 응용프로그램은 전혀 영향을 받지 않아야
- 보안
  - 데이터베이스 안의 데이터에 데이터를 소유한 사람이나 데이터에 접근이 허가된 사람만 접근할 수 있어야 함
  - 접근할 때도 사용자의 계정에 따라서 다른 권한 가짐

DBMS 개요

DB/DBMS의 특징

- 데이터 중복의 최소화
  - 동일한 데이터가 여러 개 중복되어 저장되는 것 방지
- 응용프로그램 제작 및 수정이 쉬워짐
  - 통일된 방식으로 응용프로그램 작성 가능
  - 유지보수 또한 쉬워짐
- 데이터의 안전성 향상
  - 대부분의 DBMS가 제공하는 백업·복원 기능 이용
  - 데이터가 깨지는 문제가 발생할 경우 원상으로 복원, 복구하는 방법이 명확해짐

DBMS 개요

데이터베이스의 발전

- 파일시스템 사용
  - 컴퓨터 파일에 기록/저장 – 메모장, 엑셀 활용
  - 컴퓨터에 저장된 파일의 내용은 읽고, 쓰기가 편한 약속된 형태의 구조 사용
  - 데이터의 양이 많아지면 데이터 중복으로 인한 불일치 위험

DBMS 개요

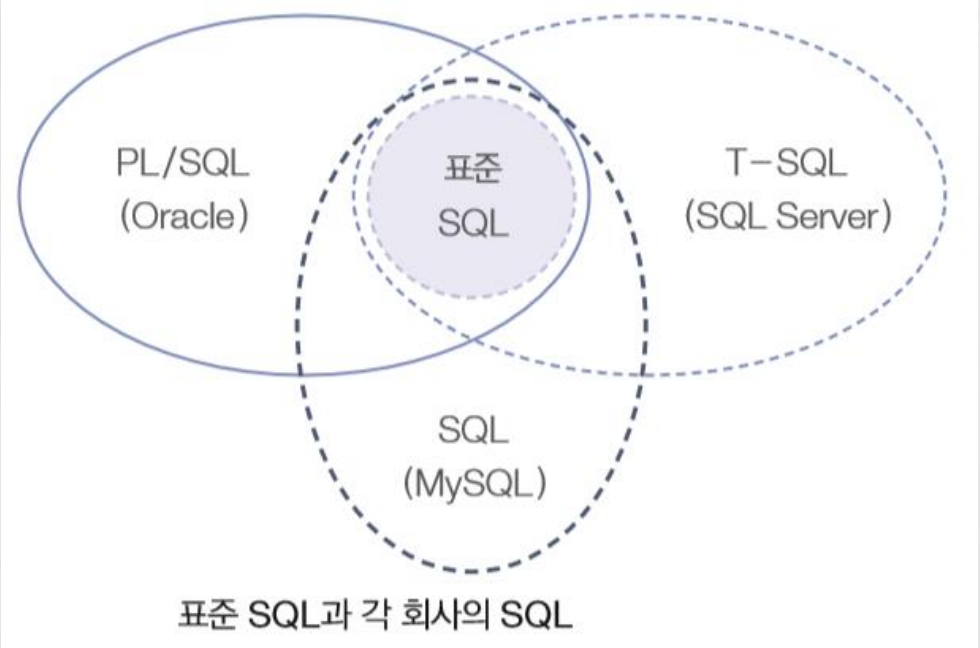
데이터베이스의 발전

- 데이터베이스 관리시스템
  - 파일시스템의 단점 보완
  - 대량의 데이터를 보다 효율적으로 관리하고 운영하기 위해 사용
  - DBMS - DataBase Management System
  - 데이터의 집합인 '데이터베이스'를 잘 관리하고 운영하기 위한 시스템 또는 소프트웨어
- SQL(Structured Query Language)
  - DBMS에 데이터 구축/관리/활용 위해서 사용되는 언어
  - DBMS를 통해 중요한 정보들을 입력, 관리, 추출

DBMS 개요

SQL 개요

- SQL (Structured Query Language)
  - 관계형 데이터베이스에서 사용되는 언어, '에스큐엘' 또는 '시퀄'
  - DBMS 제작 회사와 독립적
  - 다른 시스템으로 이식성이 좋음
  - 표준이 계속 발전중
  - 대화식 언어
  - 분산형 클라이언트/서버 구조



표준 SQL과 각 회사의 SQL

요구사항 분석과 시스템 설계 및 모델링

정보시스템 구축 절차 요약

- 분석, 설계, 구현, 시험, 유지보수의 5가지 단계
- 분석
  - 구현하고자 하는 프로젝트의 가장 첫 번째 단계
  - 시스템 분석 또는 요구사항 분석이라고 불림
  - 요구사항 분석은 현재 우리가 '무엇을(What)' 할 것인지 결정
  - 사용자의 인터뷰와 업무 조사 등을 수행
  - 분석의 결과로 많은 문서 작성
- 설계
  - 시스템 설계 또는 프로그램 설계
  - 구축하고자 하는 시스템을 '어떻게(How)' 할 것인지 결정
  - 대부분의 프로젝트에서 분석과 설계의 과정이 전체 공정의 50% 이상 차지

요구사항 분석과 시스템 설계 및 모델링

데이터베이스 모델링과 필수 용어

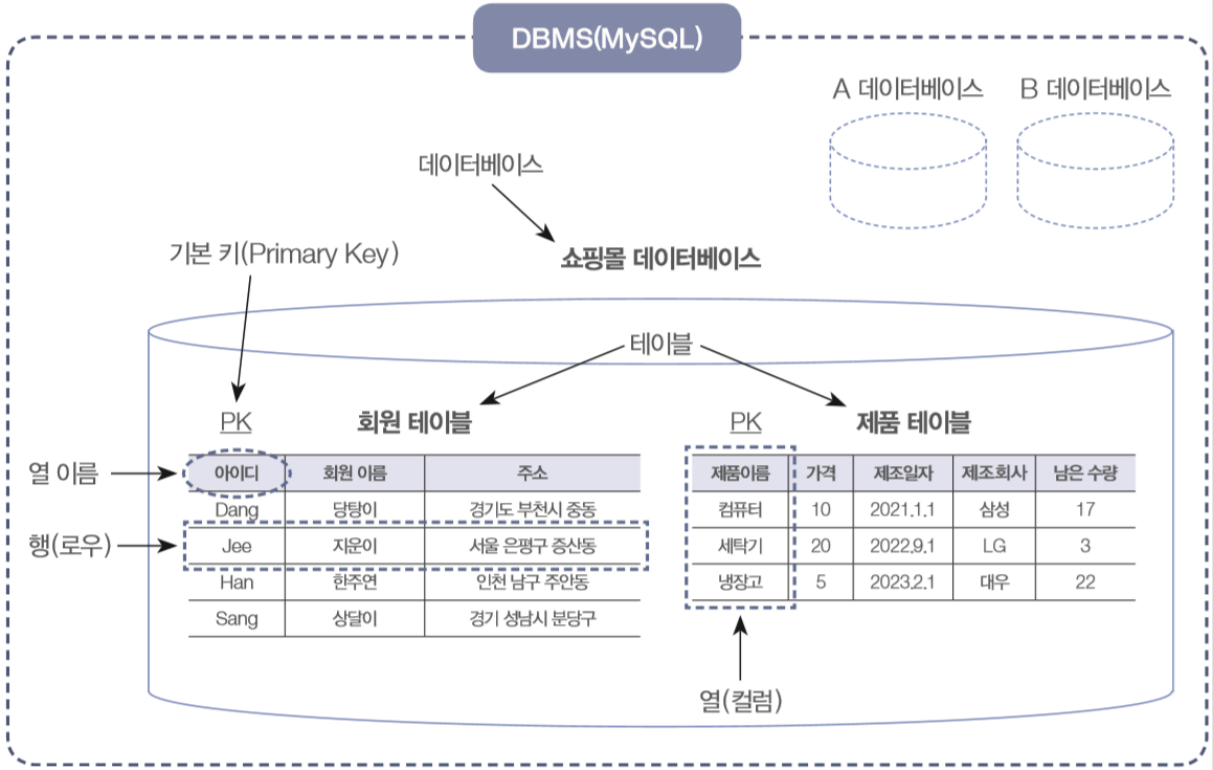
- 데이터베이스 모델링
- 현실세계에서 사용되는 데이터를 MySQL에 어떻게 옮겨 놓을 것인지를 결정하는 과정
- 저장할 정보는 테이블(Table)이라는 형식에 맞춰 저장

요구사항 분석과 시스템 설계 및 모델링

데이터베이스 모델링과 필수 용어

- 데이터베이스 모델링

- Ex) 쇼핑몰 데이터 베이스의 예



테이블의 구조와 관련 용어

요구사항 분석과 시스템 설계 및 모델링

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

요구사항 분석과 시스템 설계 및 모델링

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

요구사항 분석과 시스템 설계 및 모델링

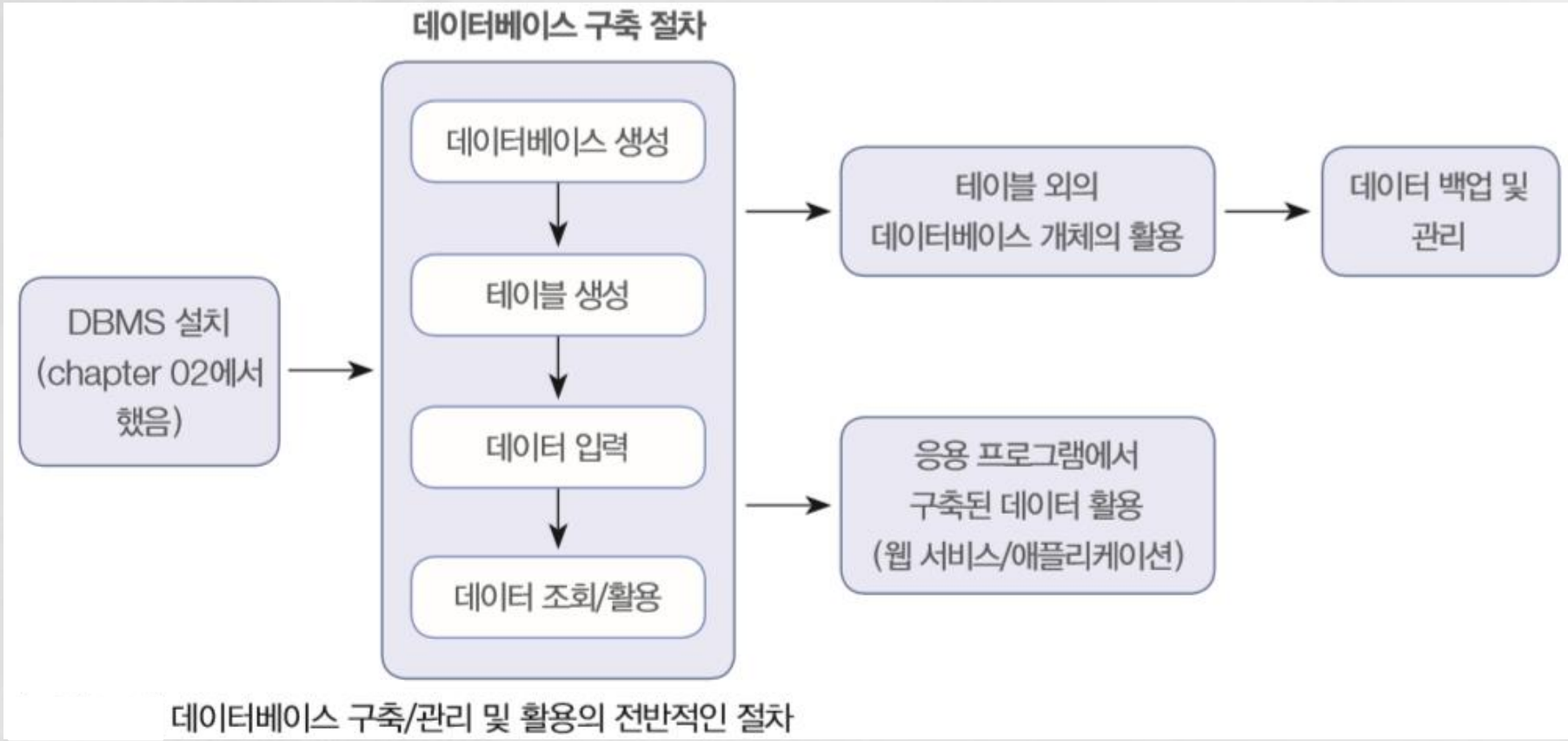
데이터베이스 모델링과 필수 용어

- 기본 키 (Primary Key) 열
  - 기본 키(또는 주 키) 열은 각 행을 구분하는 유일한 열
  - 중복되어서는 안되며, 비어 있어서도 안 됨
  - 각 테이블에는 기본 키가 하나만 지정
- 외래 키(Foreign Key) 필드
  - 두 테이블의 관계를 맺어주는 키
- SQL (Structured Query Language)
  - 구조화된 질의 언어
  - 사람과 DBMS가 소통하기 위한 말(언어)

요구사항 분석과 시스템 설계 및 모델링

데이터베이스 구축/관리 및 활용의 전반적인 절차

- 데이터베이스 구축 절차



데이터베이스 구축/관리 및 활용의 전반적인 절차

요구사항 분석과 시스템 설계 및 모델링

데이터 활용

- 주로 SELECT 문 사용해 데이터 활용
- 사용할 데이터 베이스 선택
- SELECT 열 이름 FROM 테이블 이름 [WHERE 조건]
  - 모든 데이터 출력하기 (열 이름 대신 ' * ')
  - 열을 선택해 데이터 출력하기 (열 이름 나열)
  - 특정 데이터를 만족하는 데이터 출력하기 (WHERE절에 조건 입력)
- 새로운 테이블 생성
- 테이블 삭제
  - DROP TABLE 테이블 이름

요구사항 분석과 시스템 설계 및 모델링

데이터 활용

- 뷰(View)
  - 가상의 테이블
  - 실제 행 데이터를 가지고 있지 않음
- 스토어드 프로시저 (Stored Procedure)
  - SQL문을 하나로 묶어 편리하게 사용하는 기능
  - 다른 프로그래밍 언어와 같은 기능을 담당할 수도 있음
- 트리거 (Trigger)
  - 테이블에 부착되어 테이블에 INSERT나 UPDATE 또는 DELETE 작업이 발생되면 실행되는 코드

요구사항 분석과 시스템 설계 및 모델링

데이터 활용

- 백업과 복원
  - 백업
    - 현재의 데이터베이스를 다른 매체에 보관하는 작업
  - 복원
    - 데이터베이스에 문제 발생 시 다른 매체에 백업된 데이터를 이용해 원상태로 돌려놓는 작업
  - 백업과 복원은 DBA(DataBase Administrator: 데이터베이스 관리자)가 해야 할 가장 중요한 일

SQL 기본

SELECT문

<SELECT... FROM>

- 원하는 데이터를 가져와 주는 기본적인 구문
- 가장 많이 사용되는 구문
- 데이터베이스 내 테이블에서 원하는 정보 추출하는 명령

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

```
SELECT 열 이름
FROM 테이블이름
WHERE 조건
```

SELECT문

USE 구문

- SELECT문 학습 위해 사용할 데이터베이스 지정
- 지정해 놓은 후 특별히 다시 USE문 사용하거나 다른 DB를 사용하겠다고 명시하지 않는 이상 모든 SQL문은 지정 DB에서 수행

```
USE 데이터베이스_이름;
```

- employees를 사용하기 위해서는 쿼리 창에 입력

```
USE employees;
```

SELECT문

SELECT와 FROM

- SELECT *
  - 선택된 DB가 employees 라면 다음 두 쿼리는 동일

```
SELECT * FROM employees.titles;
SELECT * FROM titles;
```

- SELECT 열 이름
  - 테이블에서 필요로 하는 열만 가져오기 가능

```
SELECT first_name FROM employees;
```

  - 여러 개의 열을 가져오고 싶을 때는 콤마로 구분

```
SELECT first_name, last_name, gender FROM employees;
```

  - 열 이름의 순서는 출력하고 싶은 순서대로 배열 가능

SELECT문

SELECT와 FROM

- 주석(Remark)
  - -- 한 줄 수석

```
SELECT first_name, last_name, gender  -- 이름과 성별 열을 가져옴
FROM employees;
```

  - 여러 줄 주석 /* */

```
/* 블록 주석
SELECT first_name, last_name, gender
FROM employees;
*/
```

SELECT문

DB, TABLE, 열의 이름이 확실하지 않을 때 조회하는 방법

- 현재 서버에 어떤 DB가 있는지 보기
  - SHOW DATABASES;
- 현재 서버에 어떤 TABLE이 있는지 보기
  - 데이터베이스에 있는 테이블 정보 조회
    - SHOW TABLE STATUS;
  - 테이블 이름만 간단히 보기
    - SHOW TABLES;
- employees 테이블의 열이 무엇이 있는지 확인
  - DESCRIBE employees; 또는 DESC employees;

SELECT문

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

- 기본적인 WHERE절
  - 조회하는 결과에 특정한 조건을 줘서 원하는 데이터만 보고 싶을 때 사용
  - SELECT 필드이름 FROM 테이블이름 WHERE 조건식;

```
SELECT * FROM usertbl WHERE name = '김경호';
```

- 관계 연산자의 사용
  - OR 연산자 : '...했거나', '... 또는'
  - AND 연산자 : '...하고', '...면서', '... 그리고'
  - 조건 연산자(=, <, >, <=, >=, < >, != 등)와 관계 연산자(NOT, AND, OR 등)를 조합하여 데이터를 효율적으로 추출 가능

```
SELECT userID, Name FROM usertbl WHERE birthYear >= 1970 AND height >= 182;
```

SELECT문

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

- BETWEEN... AND와 IN( ) 그리고 LIKE
  - 데이터가 숫자로 구성되어 있으며 연속적인 값 : BETWEEN ... AND 사용

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

- 이산적인(Discrete) 값의 조건 : IN() 사용

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

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

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

SELECT문

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');
```

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

SELECT문

원하는 순서대로 정렬하여 출력 : ORDER BY

- ORDER BY절
  - 결과물에 대해 영향을 미치지는 않고 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

SELECT문

원하는 순서대로 정렬하여 출력 : ORDER BY

- 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

SELECT문

원하는 순서대로 정렬하여 출력 : ORDER BY

- 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

SELECT문

GROUP BY 및 HAVING 그리고 집계 함수

- GROUP BY절
  - 그룹으로 묶어주는 역할
  - 집계 함수(Aggregate Function)와 함께 사용
    - 효율적인 데이터 그룹화 (Grouping)
    - 각 사용자 별로 구매한 개수를 합쳐 출력

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

    - 읽기 좋게 하기 위해 별칭(Alias) AS 사용

```
SELECT userID AS '사용자 아이디', SUM(amount) AS '총 구매 개수'
   FROM buytbl GROUP BY userID;
```

SELECT문

GROUP BY 및 HAVING 그리고 집계 함수

- GROUP BY와 함께 자주 사용되는 집계 함수

함수명	설명
AVG( )	평균을 구한다.
MIN( )	최소값을 구한다.
MAX( )	최대값을 구한다.
COUNT( )	행의 개수를 센다.
COUNT(DISTINCT)	행의 개수를 센다(중복은 1개만 인정).
STDEV( )	표준편차를 구한다.
VAR_SAMP( )	분산을 구한다.

GROUP BY와 함께 사용되는 집계 함수

- 전체 구매자가 구매한 물품의 개수 평균

```
USE sqldb;
SELECT AVG(amount) AS '평균 구매 개수' FROM buytbl ;
```

SELECT문

GROUP BY 및 HAVING 그리고 집계 함수

- Having절
  - WHERE와 비슷한 개념으로 조건 제한하는 것이지만, 집계 함수에 대해서 조건을 제한하는 것
  - HAVING절은 꼭 GROUP BY절 다음에 나와야 함(순서 바뀌면 안됨)
- ROLLUP
  - 총합 또는 중간 합계가 필요할 경우 사용
  - GROUP BY절과 함께 WITH ROLLUP문 사용

SELECT문

GROUP BY 및 HAVING 그리고 집계 함수

- ROLLUP
  - 총합 또는 중간 합계가 필요할 경우 사용
  - GROUP BY절과 함께 WITH ROLLUP문 사용
  - ex) 분류(groupName) 별로 합계 및 그 총합 구하기

```
SELECT num, groupName, SUM(price * amount) AS '비용'
    FROM buytbl
    GROUP BY  groupName, num
    WITH ROLLUP;
```

SELECT문

SQL의 분류

- DML (Data Manipulation Language, 데이터 조작 언어)
  - 데이터를 조작(선택, 삽입, 수정, 삭제)하는 데 사용되는 언어
  - DML 구문이 사용되는 대상은 테이블의 행
  - DML 사용하기 위해서는 테이블이 정의되어 있어야 함
  - SQL문 중 SELECT, INSERT, UPDATE, DELETE가 이 구문에 해당
  - 트랜잭션(Transaction)이 발생하는 SQL도 DML에 속함
    - 테이블의 데이터를 변경(입력/수정/삭제)할 때 실제 테이블에 완전히 적용하지 않고, 임시로 적용시키는 것
    - 취소 가능

SELECT문

SQL의 분류

- DDL (Data Definition Language, 데이터 정의 언어)
  - 데이터베이스, 테이블, 뷰, 인덱스 등의 데이터베이스 개체를 생성/삭제/변경하는 역할
  - CREATE, DROP, ALTER 자주 사용
  - DDL은 트랜잭션 발생시키지 않음
  - 되돌림(ROLLBACK)이나 완전적용(COMMIT) 사용 불가
- DCL (Data Control Language, 데이터 제어 언어)
  - 사용자에게 어떤 권한을 부여하거나 빼앗을 때 주로 사용하는 구문
  - GRANT/REVOKE/DENY 구문

데이터의 변경을 위한 SQL문

데이터의 삽입 : INSERT

- INSERT문의 기본
  - 테이블 이름 다음에 나오는 열 생략 가능

```
INSERT [INTO] 테이블[(열1, 열2, …)] VALUES (값1, 값2 …)
```

  - 생략할 경우에 VALUES 다음에 나오는 값들의 순서 및 개수가 테이블이 정의된 열 순서 및 개수와 동일해야 함

데이터의 변경을 위한 SQL문

데이터의 삽입 : INSERT

- 자동으로 증가하는 AUTO_INCREMENT
  - INSERT에서는 해당 열이 없다고 생각하고 입력
    - INSERT문에서 NULL 값 지정하면 자동으로 값 입력
  - 1부터 증가하는 값 자동 입력
  - 적용할 열이 PRIMARY KEY 또는 UNIQUE일 때만 사용가능
  - 데이터 형은 숫자 형식만 사용 가능

데이터의 변경을 위한 SQL문

데이터의 삽입 : INSERT

- 대량의 샘플 데이터 생성
  - INSERT INTO … SELECT 구문 사용

```
형식:
INSERT INTO 테이블이름 (열 이름1, 열 이름2, …)
   SELECT문  ;
```

  - 다른 테이블의 데이터를 가져와 대량으로 입력하는 효과
  - SELECT문의 열의 개수 = INSERT 할 테이블의 열의 개수
  - 테이블 정의 까지 생략 하려면 CREATE TABLE … SELECT 구문을 사용

데이터의 변경을 위한 SQL문

데이터의 수정 : UPDATE

- 기존에 입력되어 있는 값 변경하는 구문

```
UPDATE 테이블이름
    SET 열1=값1, 열2=값2 …
    WHERE 조건 ;
```

- WHERE절 생략 가능하나 WHERE절 생략하면 테이블의 전체 행의 내용 변경
  - 실무에서 실수가 종종 일어남, 주의 필요
  - 원상태로 복구하기 복잡하며, 다시 되돌릴 수 없는 경우도 있음

데이터의 변경을 위한 SQL문

데이터의 삭제 : DELETE FROM

- 행 단위로 데이터 삭제하는 구문

```
DELETE FROM 테이블이름 WHERE 조건;
```

- WHERE절 생략되면 전체 데이터를 삭제함
- 테이블을 삭제하는 경우의 속도 비교
  - DML문인 DELETE는 트랜잭션 로그 기록 작업 때문에 삭제 느림
  - DDL문인 DROP과 TRUNCATE문은 트랜잭션 없어 빠름
    - 테이블 자체가 필요 없을 경우에는 DROP 으로 삭제
    - 테이블의 구조는 남겨놓고 싶다면 TRUNCATE로 삭제하는 것이 효율적

데이터의 변경을 위한 SQL문

조건부 데이터 입력, 변경

- 기본 키가 중복된 데이터를 입력한 경우
  - 오류로 입력 불가
- 대용량 데이터 처리의 경우 에러 발생하지 않은 구문 실행
  - INSERT IGNORE문
    - 에러 발생해도 다음 구문으로 넘어가게 처리
    - 에러 메시지 보면 적용되지 않은 구문이 어느 것인지 구분 가능
  - ON DUPLICATE KEY UPDATE 구문
    - 기본 키가 중복되면 데이터를 수정되도록 하는 구문도 활용 가능

정리

정리

- DBMS 개념
- 데이터베이스 모델링
- SQL 기본
- SELECT
- UPDATE
- INSERT
- DELETE