MySql

1.1 DBMS 개요

❖ 데이터베이스의 정의와 특징

▪ 데이터베이스

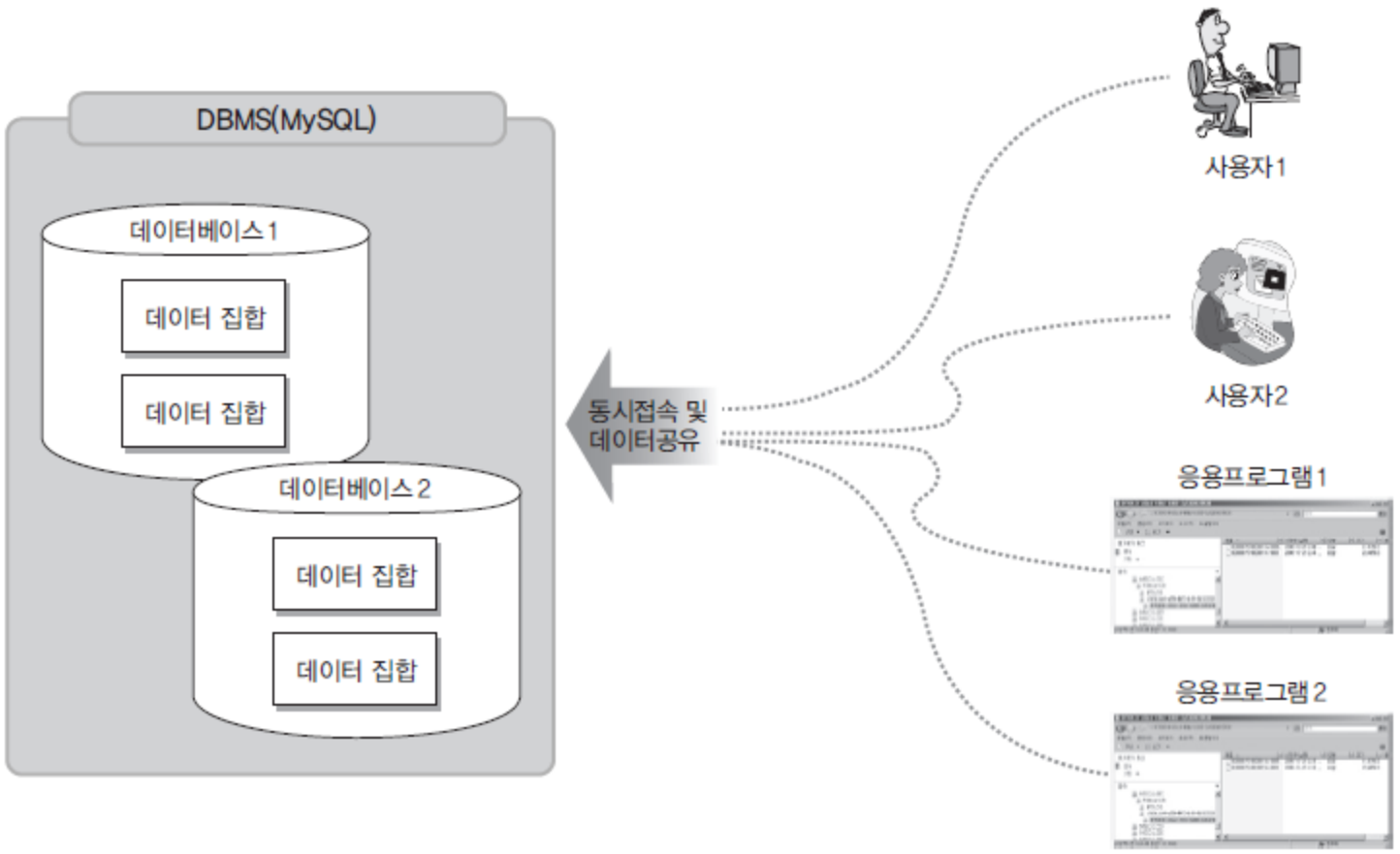
- ‘데이터의 집합’
- 여러 명의 사용자나 응용프로그램이 공유하는 데이터들
- 동시에 접근 가능해야
- ‘데이터의 저장 공간’ 자체

▪ DBMS

- 데이터베이스를 관리·운영하는 역할

1.1 DBMS 개요

❖ DBMS 개념도



1.1 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
  - **기존에 작성된 응용프로그램은 전혀 영향을 받지 않아야**

▪ 보안

- 데이터베이스 안의 데이터에 데이터를 소유한 사람이나 데이터에 접근이 허가된 사람만 접근할 수 있어야
- 접근할 때도 사용자의 계정에 따라서 다른 권한 가짐

1.1 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

1.1 DBMS 개요

❖ 데이터베이스의 발전

▪ 오프라인 관리

- 종이에 연필로 기록해 장부로 관리

1.1 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)

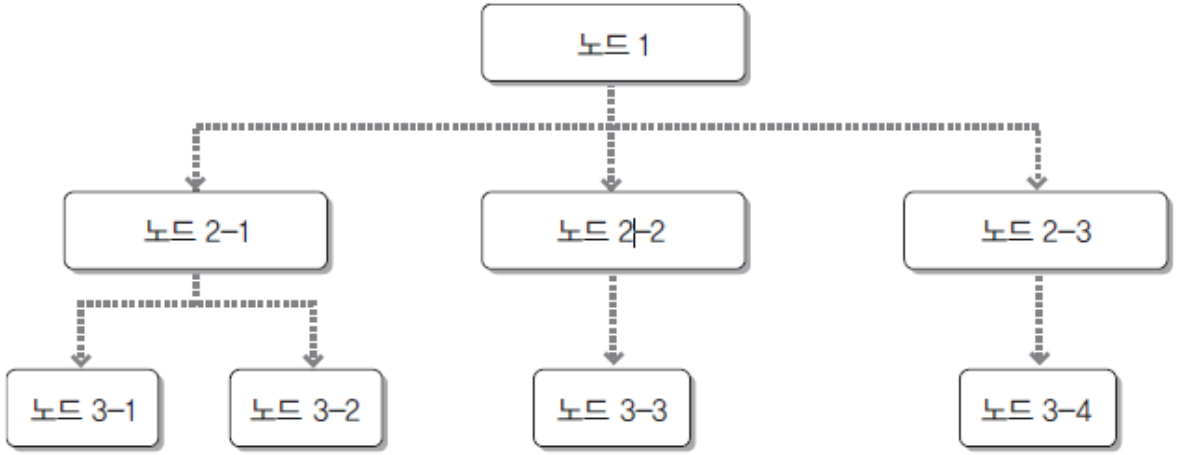
- DBMS에 데이터 구축/관리/활용 위해서 사용되는 언어
- DBMS를 통해 중요한 정보들을 입력, 관리, 추출

1.1 DBMS 개요

❖ DBMS 분류

▪ 계층형 DBMS

- 처음으로 나온 DBMS 개념 – 1960년대에 시작
- 각 계층은 트리Tree 형태, 1:N 관계
- 문제점
  - **처음 구축한 이후 그 구조를 변경하기가 상당히 까다로움**
  - **주어진 상태에서의 검색은 상당히 빠름**
  - **접근 유연성 부족해서 임의의 검색에는 어려움**

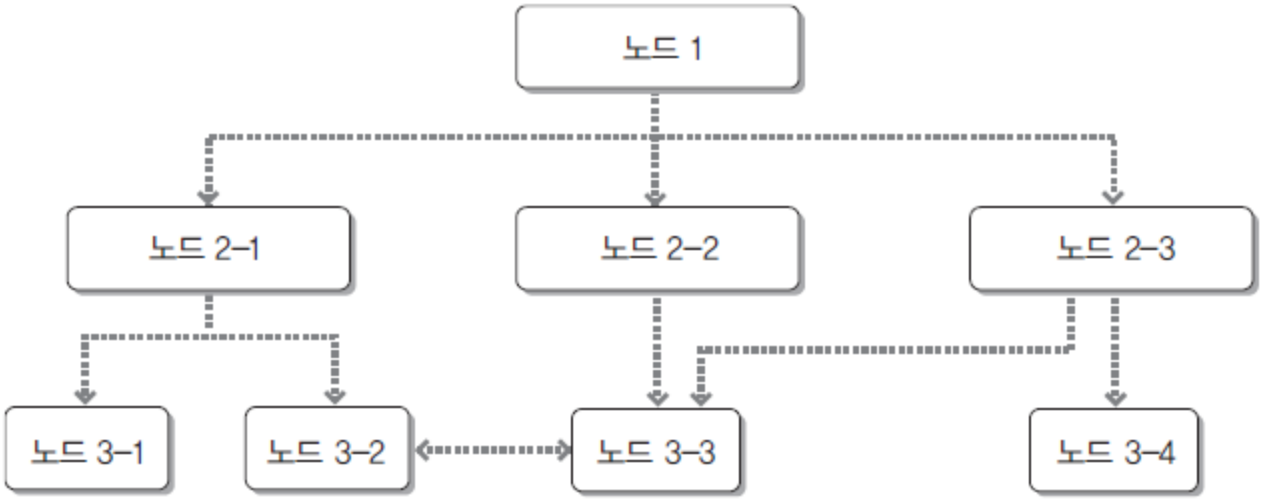


1.1 DBMS 개요

❖ DBMS 분류

▪ 망형 DBMS

- 계층형 DBMS의 문제점을 개선하기 위해 1970년대에 시작
- 1:1,1:N, N:M(다대다) 관계 지원 – 효과적이고 빠른 데이터 추출
- 복잡한 내부 포인터 사용
  - **프로그래머가 이 모든 구조를 이해해야만 프로그램의 작성 가능**

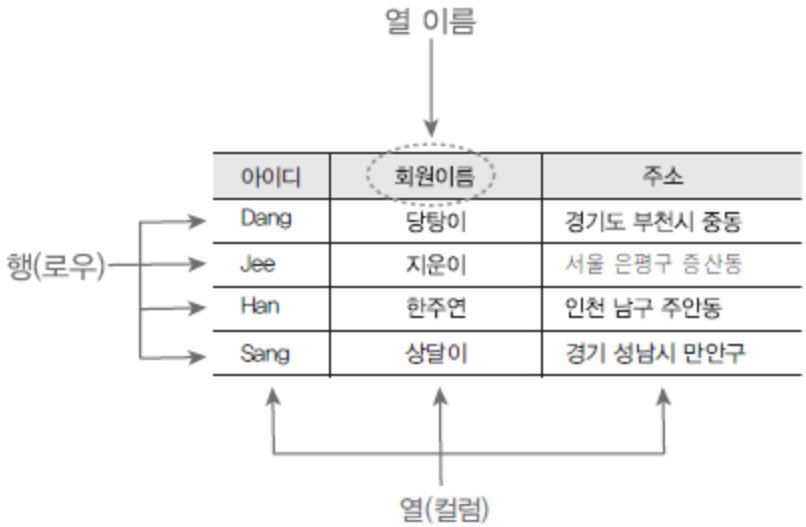


1.1 DBMS 개요

❖ DBMS 분류

▪ 관계형 DBMS (Relational DBMS)

- 1969년 E.F.Codd라는 학자가 수학 모델에 근거해 고안
- 데이터베이스는 테이블Table이라 불리는 최소 단위로 구성
- 이 테이블은 하나 이상의 열로 구성



1.1 DBMS 개요

❖ 관계형 DBMS (Relational DBMS)의 장단점

▪ 장점

- 다른 DBMS에 비해 업무가 변화될 경우 쉽게 변화에 순응
- 유지보수 측면에서도 편리
- 대용량 데이터의 관리와 데이터 무결성Integration보장

▪ 단점

- 시스템 자원을 많이 차지해 시스템이 전반적으로 느려지는 것
  - **하드웨어 발전되어 해결**

1.1 DBMS 개요

❖ SQL 개요 (p.12~13)

- **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**

2 . 1 요구사항 분석과 시스템 설계 그리고 모델링

❖ 정보시스템 구축 절차 요약

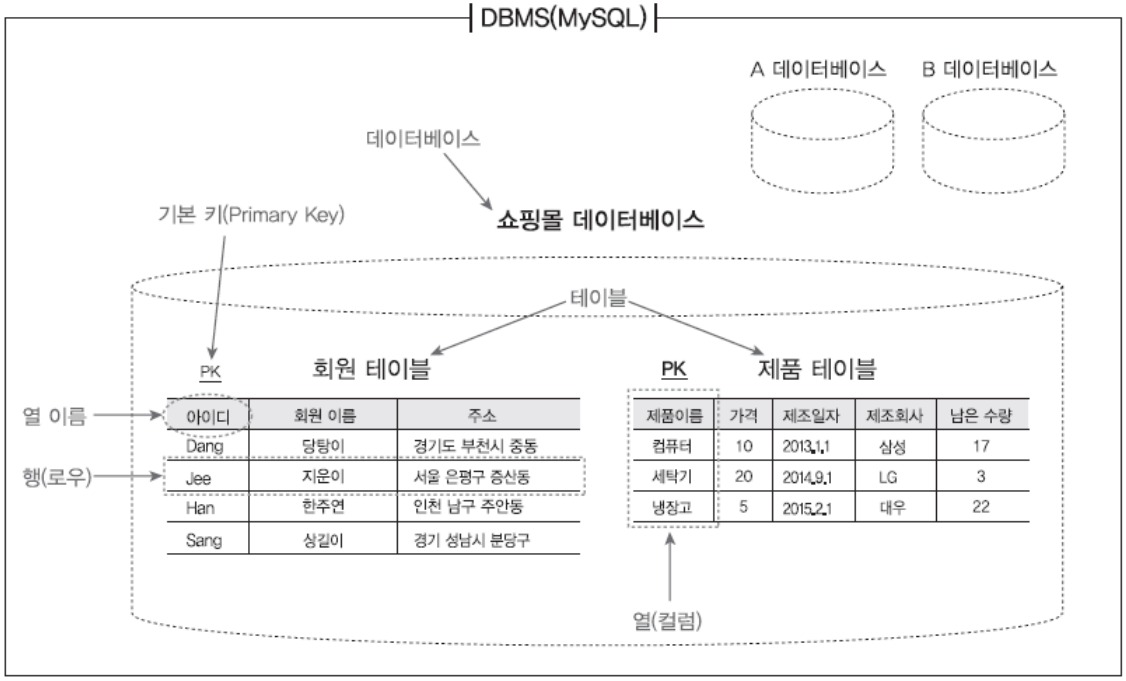
- **분석, 설계, 구현, 테스트, 유지보수의 5가지 단계**
- **분석**
  - 구현하고자 하는 프로젝트의 가장 첫 번째 단계
  - 시스템 분석 또는 요구사항 분석이라고 불림
  - 요구사항 분석은 현재 우리가 '무엇을(What)' 할 것인지 결정
  - 사용자의 인터뷰와 업무 조사 등을 수행
  - 프로젝트의 첫 단추를 끼우는 중요한 단계
  - 분석의 결과로 많은 문서 작성
- **설계**
  - 시스템 설계 또는 프로그램 설계
  - 우리가 구축하고자 하는 시스템을 '어떻게(How)' 할 것인지 결정
  - 대부분의 프로젝트에서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

2 . 1 요구사항 분석과 시스템 설계 그리고 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
- Ex) 쇼핑몰 데이터 베이스의 예



2 . 1 요구사항 분석과 시스템 설계 그리고 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

2. 1 요구사항 분석과 시스템 설계 그리고 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

2.1 요구사항 분석과 시스템 설계 그리고 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
- 4장 이후 설명

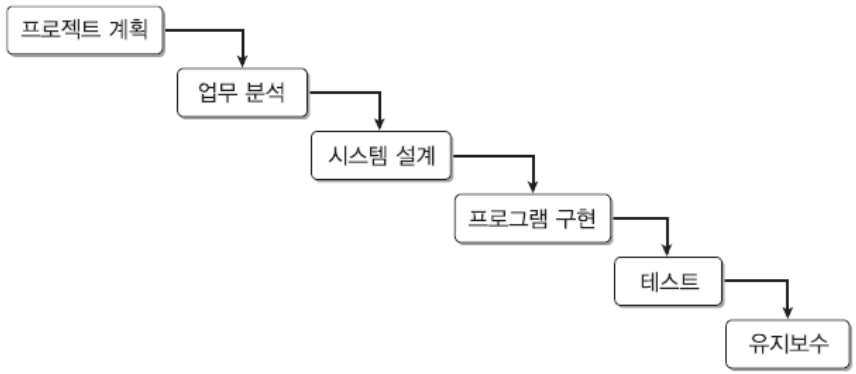
▪ SQL (Structured Query Language)

- 구조화된 질의 언어
- 사람과 DBMS가 소통하기 위한 말(언어)
- 6, 7장에서 자세히 다룸

3 . 1 프로젝트의 진행 단계

❖ 프로젝트 (Project)

- **‘현실세계의 업무를 컴퓨터 시스템으로 옮겨놓는 일련의 과정’**
- **‘대규모의 프로그램을 작성하기 위한 전체 과정’**
  - Ex) 집 짓기의 경우 초가집 → 목조건물→ 수 십층 이상의 건물
- **‘소프트웨어 개발 방법론’ 의 대두**
- **폭포수 모델 (Waterfall Model)**



3.1 프로젝트의 진행 단계

❖ 폭포수 모델 (Waterfall Model)

▪ 가장 오래되고 전통적으로 사용되는 소프트웨어 개발 모델

- 폭포가 떨어지듯이 각 단계가 끝나면 다음 단계로 진행

▪ 장점

- 각 단계가 명확히 구분되어 프로젝트의 진행 단계가 명확해짐

▪ 단점

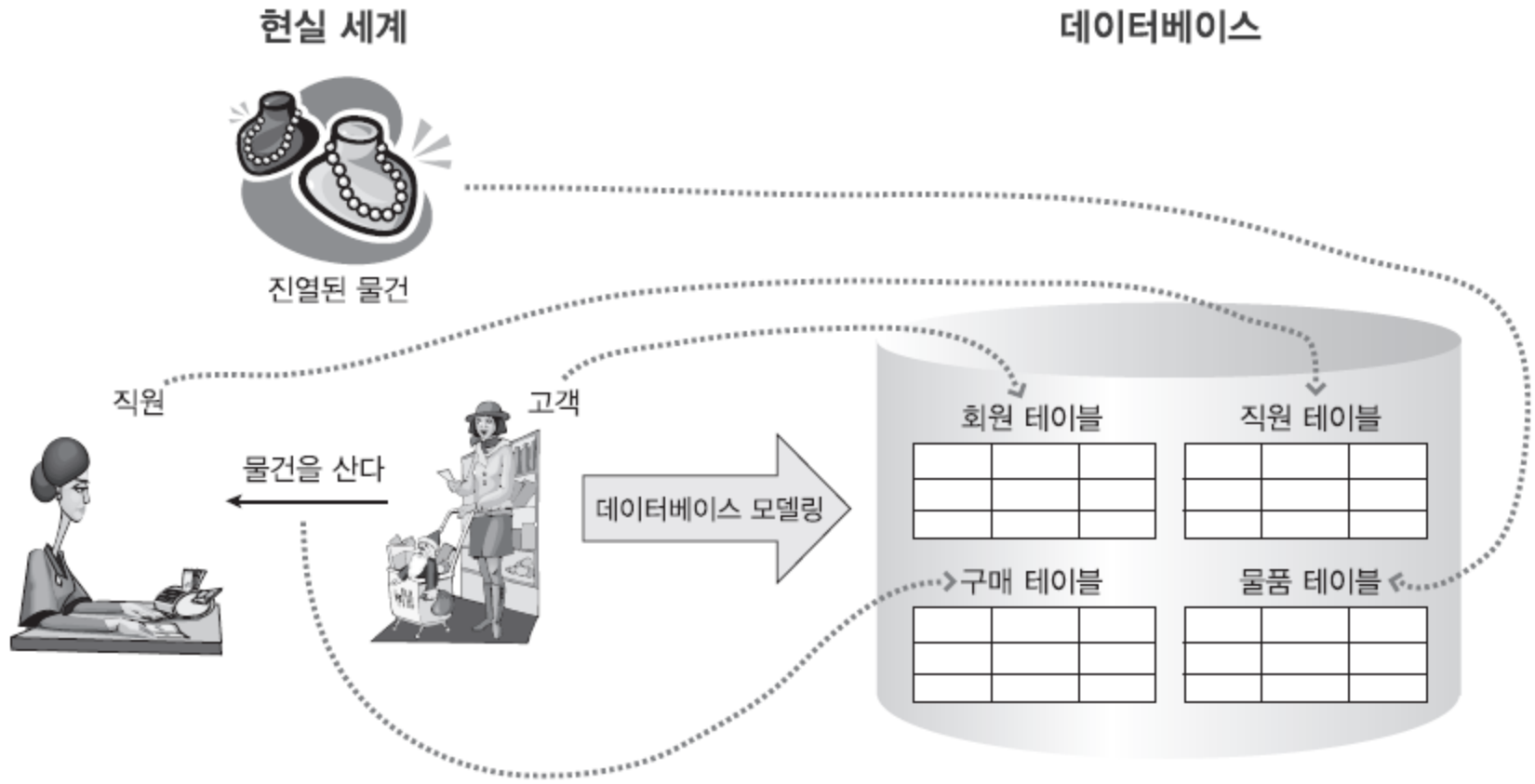
- 문제점이 발생될 경우 다시 앞 단계로 거슬러 올라가기가 어려움
- 문제점이 대부분 프로그램 구현 단계나 테스트 단계에서 발생
- 해결은 업무 분석단계에서 다시 시작
  - **업무 분석과 시스템 설계에 50% 이상 할당**

3 . 2 데이터베이스 모델링

❖ 데이터베이스 모델링(데이터 모델링)

- 현 세계에서 사용되는 작업이나 사물들

→ DBMS의 데이터베이스 개체로 옮기기 위한 과정



4 . 1 SELECT문

- <SELECT… FROM>
  - 원하는 데이터를 가져와 주는 기본적인 구문
  - 가장 많이 사용되는 구문
  - 데이터베이스 내 테이블에서 원하는 정보 추출하는 기능

4 . 1 SELECT문

❖ SELECT의 구문 형식

- p.183의 복잡한 형식을 실제 사용되는 형태로 요약

```
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 열이름
FROM 테이블이름
WHERE 조건
```

4. 1 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;
```

- Workbench 에서 직접 선택해서 사용도 가능
  - [Navigator]의 [Schemas] 탭
    - employees 데이터베이스를 더블 클릭하면 진한 글자 전환
    - 왼쪽 아래 'Active schema changed to employees' 메시지

4 . 1 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
- 여러 개의 열을 가져오고 싶을 때는 콤마로 구분
- 열 이름의 순서는 출력하고 싶은 순서대로 배열 가능

4 . 1 SELECT문

❖ 특정 조건의 데이터만 조회 - <SELECT FROM WHERE>

▪ 기본적인 WHERE절

- 조회하는 결과에 특정한 조건 줘서 원하는 데이터만 보고 싶을 때 사용
- SELECT 필드이름 FROM 테이블이름 WHERE 조건식;
- 조건이 없을 경우 테이블의 크기가 클수록 찾는 시간과 노력이 증가

▪ 관계 연산자의 사용

- '…했거나', '… 또는' - OR 연산자
- '…하고', '…면서', '… 그리고' - AND 연산자
- 조건 연산자(=, <, >, <=, >=, < >, != 등)와 관계 연산자(NOT, AND, OR 등)의 조합으로 알맞은 데이터를 효율적으로 추출

4 . 1 SELECT문

❖ 특정 조건의 데이터만 조회 – <SELECT FROM WHERE>

▪ BETWEEN… AND와 IN( ) 그리고 LIKE

- 데이터가 숫자로 구성되어 있어 연속적인 값
  - **BETWEEN… AND 사용 가능**

- 이산적인 (Discrete) 값의 조건
  - **IN( )**
  - **Ex) SELECT Name, addr FROM userTbl WHERE addr= '경남' OR addr= '전남' OR addr= '경북';**
    - » **SELECT Name, addr FROM userTbl WHERE addr IN ('경남','전남','경북');**

- 문자열의 내용 검색하기 위해 LIKE 연산자 사용
  - **문자 뒤에 % – 무엇이든(%) 허용**
  - **한 글자와 매치하기 위해서는 '_' 사용**

4 . 1 SELECT문

❖ ANY/ALL/SOME ,서브쿼리(SubQuery, 하위쿼리)

▪ 서브쿼리

- 쿼리문 안에 또 쿼리문이 들어 있는 것
- 서브쿼리 사용하는 쿼리로 변환 예제
  - **김경호보다 키가 크거나 같은 사람의 이름과 키 출력**
    - » **WHERE 조건에 김경호의 키를 직접 써줘야 함**
  - **SELECT Name, height FROM userTBL WHERE height > 177;**
  - **SELECT Name, height FROM userTbl WHERE height > (SELECT height FROM userTbl WHERE Name = '김경호');**
  - **서브쿼리의 결과가 둘 이상이 되면 에러 발생**

4 . 1 SELECT문

❖ ANY/ALL/SOME ,서브쿼리(SubQuery, 하위쿼리)

▪ ANY 구문의 필요성

- ANY
  - **서브쿼리의 여러 개의 결과 중 한 가지만 만족해도 가능**
  - **SOME은 ANY와 동일한 의미로 사용**
  - **= ANY 구문은 IN과 동일한 의미**

- ALL – 서브쿼리의 여러 개의 결과를 모두 만족시켜야 함

4 . 1 SELECT문

❖ 원하는 순서대로 정렬하여 출력 : ORDER BY

▪ ORDER BY절

- 결과물에 대해 영향을 미치지는 않음
- 결과가 출력되는 순서를 조절하는 구문
- 기본적으로 오름차순 (ASCENDING) 정렬
- 내림차순 (DESCENDING) 으로 정렬
  - **열 이름 뒤에 DESC 적어줄 것**
- ORDER BY 구문을 혼합해 사용하는 구문도 가능
  - **SELECT Name, height FROM userTbl ORDER BY height DESC, name ASC;**
  - **키가 큰 순서로 정렬하되 만약 키가 같을 경우 이름 순으로 정렬**
  - **ASC(오름차순)는 디폴트 값이므로 생략**

4 . 1 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

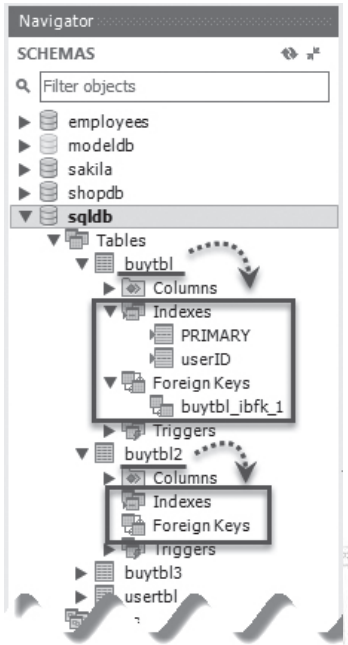
- 일부를 보기 위해 여러 건의 데이터를 출력하는 부담 줄임
- 상위의 N개만 출력하는 ‘LIMIT N’ 구문 사용
- 서버의 처리량을 많이 사용해 서버의 전반적인 성능을 나쁘게 하는 악성 쿼리문 개선할 때 사용

4 . 1 SELECT문

❖ 원하는 순서대로 정렬하여 출력 : ORDER BY

▪ 테이블을 복사하는 CREATE TABLE … SELECT

- 테이블을 복사해서 사용할 경우 주로 사용
- CREATE TABLE 새로운테이블 (SELECT 복사할열 FROM 기존테이블)
- 지정된 일부 열만 테이블로 복사하는 것도 가능
- PK나 FK 같은 제약 조건은 복사되지 않음
  - **Workbench의 [Navigator]에서 확인 가능**



4 . 1 SELECT문

❖ GROUP BY 및 HAVING 그리고 집계 함수

▪ GROUP BY절

- 말 그대로 그룹으로 묶어주는 역할
- 집계 함수 (Aggregate Function) 함께 사용
  - **효율적인 데이터 그룹화 (Grouping)**
  - **Ex) 각 사용자 별로 구매한 개수를 합쳐 출력**
- 읽기 좋게 하기 위해 별칭 (Alias) 사용

	userID	SUM(amount)
▶	BBK	19
	EJW	4
	JYP	1
	KBS	6
	SSK	5

	사용자 아이디	총 구매 개수
▶	BBK	19
	EJW	4
	JYP	1
	KBS	6
	SSK	5

	사용자 아이디	총 구매액
▶	BBK	1920
	EJW	95
	JYP	200
	KBS	1210
	SSK	75

4 . 1 SELECT문

❖ GROUP BY 및 HAVING 그리고 집계 함수

- GROUP BY와 함께 자주 사용되는 집계 함수 (집합 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.

4 . 1 SELECT문

❖ GROUP BY 및 HAVING 그리고 집계 함수

▪ Having절

- WHERE와 비슷한 개념으로 조건 제한
- 집계 함수에 대해서 조건 제한하는 편리한 개념
- HAVING절은 꼭 GROUP BY절 다음에 나와야 함 !!!

▪ ROLLUP

- 총합 또는 중간합계가 필요할 경우 사용’
- GROUP BY절과 함께 WITH ROLLUP문 사용
- Ex) 분류(groupName) 별로 합계 및 그 총합 구하

4 . 1 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 꼭 그 이전에 테이블이 정의되어 있어야 함
- SQL문 중 SELECT, INSERT, UPDATE, DELETE가 이 구문에 해당
- 트랜잭션 (Transaction)이 발생하는 SQL도 이 DML에 속함
  - **테이블의 데이터를 변경(입력/수정/삭제)할 때 실제 테이블에 완전히 적용하지 않고, 임시로 적용시키는 것**
  - **취소 가능!**

4 . 1 SELECT문

❖ SQL의 분류

▪ DDL (Data Definition Language)

- 데이터 정의 언어
- 데이터베이스, 테이블, 뷰, 인덱스 등의 데이터베이스 개체를 생성/삭제/변경하는 역할
- CREATE, DROP, ALTER 구문
- DDL은 트랜잭션 발생시키지 않음
  - **되돌림(ROLLBACK)이나 완전적용(COMMIT) 사용 불가**
  - **DDL문은 실행 즉시 MySQL에 적용**

▪ DCL (Data Control Language)

- 데이터 제어 언어
- 사용자에게 어떤 권한을 부여하거나 빼앗을 때 주로 사용하는 구문
- GRANT/REVOKE/DENY 구문

4. 2 데이터의 변경을 위한 SQL문

❖ 데이터의 삽입 : INSERT

▪ INSERT문의 기본

- 테이블 이름 다음에 나오는 열 생략 가능
  - **생략할 경우에 VALUE 다음에 나오는 값들의 순서 및 개수가 테이블이 정의된 열 순서 및 개수와 동일해야 함**

▪ 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 UNIQUE 일 때만 사용가능
- 데이터 형은 숫자 형식만 사용 가능

4. 2 데이터의 변경을 위한 SQL문

❖ 데이터의 삽입 : INSERT

▪ 대량의 샘플 데이터 생성

- INSERT INTO… SELECT 구문 사용

```
형식:
INSERT INTO 테이블이름 (열이름1, 열이름2, …)
   SELECT문  ;
```

- 다른 테이블의 데이터를 가져와 대량으로 입력하는 효과
- SELECT문의 열의 개수 = INSERT 할 테이블의 열의 개수

4. 2 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 테이블의 전체 행의 내용 변경

4. 2 데이터의 변경을 위한 SQL문

❖ 데이터의 삭제 : DELETE FROM

- **행 단위로 데이터 삭제하는 구문**
- **DELETE FROM 테이블이름 WHERE 조건;**
- **테이블을 삭제하는 경우의 속도 비교**
  - DML문인 DELETE는 트랜잭션 로그 기록 작업 때문에 삭제 느림
  - DDL문인 DROP과 TRUNCATE문은 트랜잭션 없어 빠름

4. 2 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 사용 – 에러 발생해도 다음 구문으로 넘어가게 처리
- 에러 메시지 보면 적용되지 않은 구문이 어느 것인지 구분 가능
- 기본 키가 중복되면 데이터를 수정되도록 하는 구문도 활용 가능
  - **ON DUPLICATE KEY UPDATE 구문 사용 가능**