

DATABASE

박상원

HUFS DISLab

1. Introduction

DATABASE

Chapter 1: Introduction

- 데이터베이스 시스템의 응용
- 데이터베이스 시스템과 파일 시스템
- 데이터의 관점
- 데이터 모델
- 데이터베이스 언어
- 데이터베이스 사용자와 관리자
- 트랜잭션 관리
- 데이터베이스 시스템 구조
- 응용 프로그램 구조
- 데이터베이스 시스템의 역사

Database Management System (DBMS)

- 서로 관계 있는 데이터들의 모임
- 그 데이터에 접근하기 위한 프로그램의 집합
- DBMS는 조직과 관련된 정보들은 포함
- DBMS의 목적은 사용하기에 편리하고 효율적인 환경을 제공하는 것
- Database 시스템의 응용:
 - 은행: 모든 거래
 - 항공회사: 예약 및 비행스케줄 관리
 - 대학: 학기 등록, 학년 학점 정보
 - 판매: 고객, 상품, 구매정보 관리
 - 제조업: 판매, 재고, 주문, 공급 체인망
 - 인적자원: 고용인에 대한 정보, 임금 대장, 세금 및 복지정보
- 데이터베이스는 우리 삶의 대부분에 관여

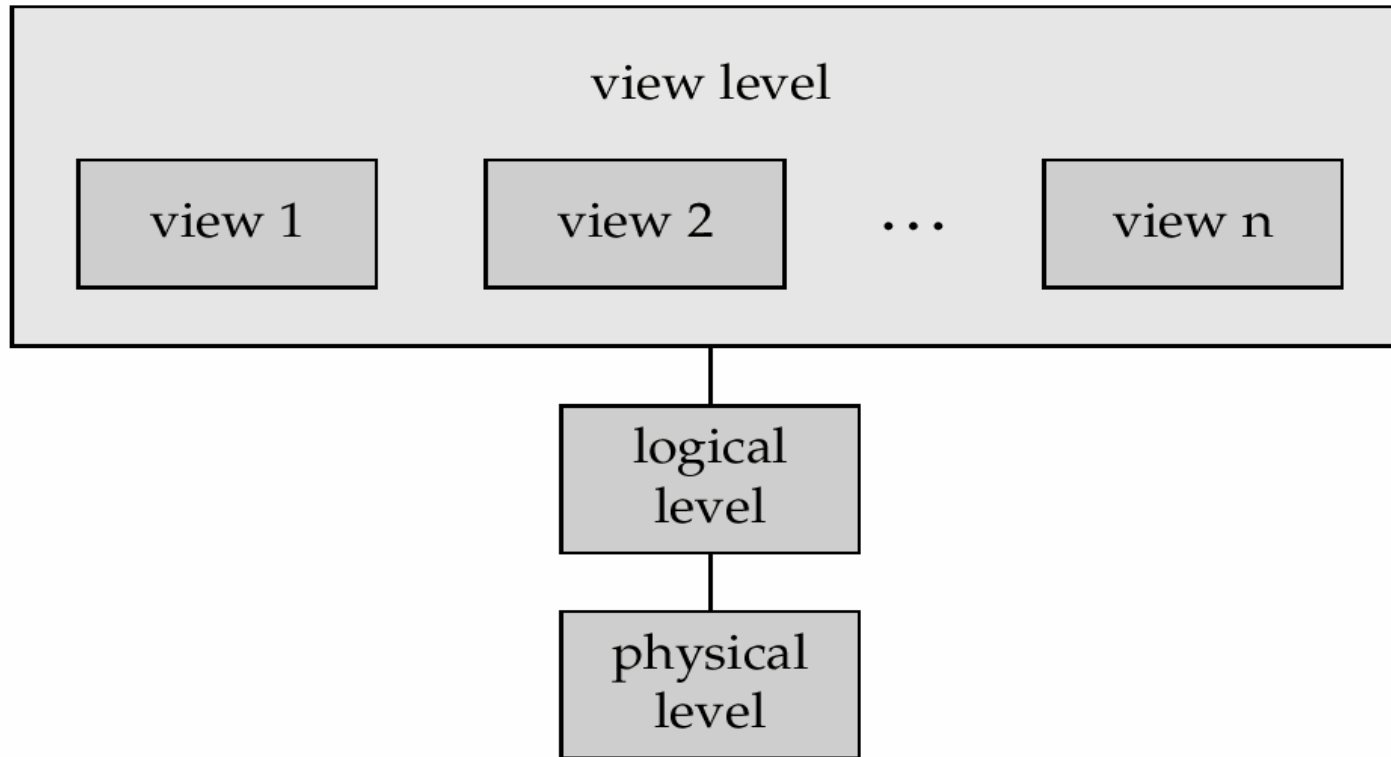
Database System의 목적 (1)

- 초기 데이터베이스 응용프로그램은 파일 시스템을 기반으로 만들어짐
- File system에 정보를 저장했을 때의 단점:
 - 데이터 중복(redundancy)과 비일관성(inconsistency)
 - 서로 다른 파일에 중복된 데이터가 저장
 - 각 파일에 저장되는 형식이 다름
 - 데이터 액세스 할 때의 난점
 - 새로운 작업이 추가될 때마다 새로운 프로그램 개발
 - 데이터의 고립(isolation)
 - 많은 파일과 서로 다른 형식
 - 무결성(integrity) 문제
 - 일관성 제약(예.계좌 잔고는 0보다 크다)을 프로그램 내에 적절한 코드를 첨가함으로써 이 조건을 잘 지키도록 해야 함
 - 기존의 프로그램을 변경하거나, 새로운 제약조건이 추가되었을 때 복잡함

Database System의 목적 (2)

- 파일 시스템에 정보를 저장했을 때의 단점(cont.)
 - 업데이트 시, 원자성(atomicity)
 - 업데이트 시, 시스템 고장이 생기면 데이터를 고장 전의 일관성 있는 상태로 유지하기 어려움
 - E.g. 계좌 A에서 계좌 B로 계좌 이체 시, 시스템 고장이 나면, 그 작업이 완료되어있거나, 아무 작업도 하지 않은 상태여야 함
 - 여러 사용자의 동시 액세스 문제
 - 성능 향상을 위해 데이터 동시 액세스가 가능해야 함
 - 동시 접근에 대한 제어가 제대로 안 될 경우, 비일관성 야기
 - E.g. 두 사람이 한 계좌에 동시에 잔고 조회와 계좌 인출을 할 경우
 - 보안 문제
- 위의 모든 문제를 해결하기 위해 데이터베이스 시스템 개발

데이터의 관점



데이터 추상화의 세 가지 단계

추상화 단계

- 물리적 단계(physical level)
 - 데이터(e.g., customer)가 어떻게 저장되는지 기술
- 논리적 단계(logical level)
 - 어떤 데이터가 저장되었는지, 데이터들 사이에 어떤 관계가 있는지 기술

```
type customer = record  
    name : string;  
    street : string;  
    city : integer;  
  
    end;
```
- 뷰 단계(view level)
 - 데이터 타입의 상세한 부분을 숨기고 있는 응용 프로그램을 사용
 - 또한, 뷰는 보안을 목적으로 정보를 숨기기 위해 사용할 수 있음

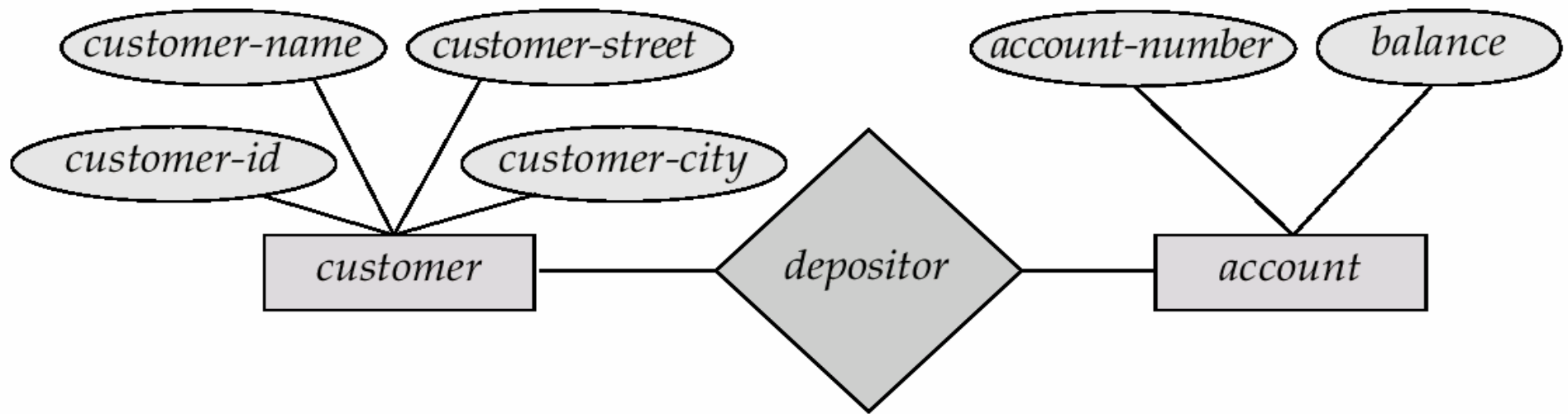
인스턴스와 스키마

- 프로그래밍 언어의 변수 타입과 비슷함
- 스키마(schema)
 - 데이터베이스의 논리적 구조
 - e.g., 데이터베이스는 customer와 account와 그 사이의 관계에 대한 정보를 포함
 - 프로그램에서 변수 선언과 유사
 - 물리적 스키마(physical schema): 물리적 단계에서 기술한 데이터베이스 설계
 - 논리적 스키마(logical schema): 논리적 단계에서 기술한 데이터베이스 설계
- 인스턴스(instance)
 - 어느 특정한 순간에 데이터베이스에 저장되어 있는 정보의 모임
 - 변수에 저장된 값과 유사
- 물리적 데이터 독립성(physical data independence)
 - 논리적 스키마의 변화 없이 물리적 스키마를 수정할 수 있는 것
 - 응용 프로그램은 논리적 스키마에 의존함
 - 일반적으로, 여러 단계의 컴포넌트 사이의 인터페이스는 다른 부분이 바뀌어도 심각한 영향이 없도록 정의되어야 함

데이터 모델

- 데이터베이스 구조의 기반
 - 데이터
 - 관계(relationship)
 - 데이터의 의미
 - 제약 조건(data constraints)
- 개체-관계 모델(entity-relationship model)
- 관계형 모델(relational model)
- 기타 데이터 모델:
 - 객체-지향 데이터 모델(object-oriented data model)
 - 반구조 데이터 모델(semi-structured data model)
 - 네티워크형 데이터 모델(network data model)
 - 계층형 데이터 모델(hierarchical data model)

개체-관계 모델 (1)

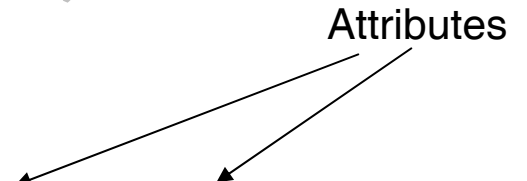


E-R 다이어그램의 예

개체-관계 모델 (2)

- 실생활의 E-R 모델
 - 개체(entity)
 - Object
 - E.g. 고객, 계좌
 - 개체 사이의 관계(relationship)
 - 예. A-101 계좌는 Johnson이라는 고객의 것
 - Depositor 관계 집합은 customer를 그가 가진 계좌(account)에 연관
- 데이터베이스 디자인에 널리 쓰임
 - E-R모델로 디자인된 데이터베이스는 보통 저장과 연산을 위한 관계 모델을 발전시킨 것

관계형 (relational model)



<i>Customer-id</i>	<i>customer-name</i>	<i>customer-street</i>	<i>customer-city</i>	<i>account-number</i>
192-83-7465	Johnson	Alma	Palo Alto	A-101
019-28-3746	Smith	North	Rye	A-215
192-83-7465	Johnson	Alma	Palo Alto	A-201
321-12-3123	Jones	Main	Harrison	A-217
019-28-3746	Smith	North	Rye	A-201

관계형 모델의 간단한 예

관계형 데이터베이스의 간단한 예

<i>customer-id</i>	<i>customer-name</i>	<i>customer-street</i>	<i>customer-city</i>
192-83-7465	Johnson	12 Alma St.	Palo Alto
019-28-3746	Smith	4 North St.	Rye
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

<i>account-number</i>	<i>balance</i>
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

<i>customer-id</i>	<i>account-number</i>
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table

데이터베이스 언어

- 데이터 정의 언어
 - Data definition language: DDL
- 데이터 조작 언어
 - Data manipulation language: DML
- SQL

데이터 정의 언어

- Data Definition Language: DDL
- 데이터베이스 스키마를 정의하기 위한 명확한 표기
 - 예)

```
create table account (  
    account-number char(10),  
    balance integer)
```
- DDL 컴파일러는 데이터 사전(data dictionary)이라는 특별한 파일에 여러 개의 테이블로 저장
- 데이터 사전은 메타데이터(metadata, 데이터에 대한 데이터)를 포함
 - 데이터베이스 스키마
 - 데이터 저장 및 정의 언어
 - 데이터베이스 시스템에 의해 사용되는 저장 구조와 액세스 방법에 대한 언어
 - 데이터 정의 언어의 확장

데이터 조작 언어

- Data Manipulation Language: DML
- 사용자가 적절한 데이터 모델로 구성된 데이터를 액세스 하거나 조작할 수 있도록 하는 언어
 - DML 또한 질의어(query language)로 잘 알려져 있음
- 두 가지 형태의 언어
 - 절차식(procedural) DML – 어떤 데이터가 필요하며 그 데이터를 어떻게 구할지에 대한 절차를 지정(How)
 - 비절차식(nonprocedural) DML – 필요한 데이터를 어떻게 구할지 명시할 필요 없이, 어떠한 데이터가 필요한지 지정하도록 사용자에게 요구함(What)
- SQL은 가장 널리 쓰이는 질의어

SQL

- SQL : 가장 널리 쓰이는 nonprocedural 언어
 - E.g. customer-id가 192-83-7465인 고객명 찾기

```
select  customer.customer-name
from    customer
where   customer.customer-id = '192-83-7465'
```
 - E.g. customer-id가 192-83-7465인 고객이 가지고 있는 모든 계좌 잔액 조회

```
select  account.balance
from    depositor, account
where   depositor.customer-id = '192-83-7465' and
        depositor.account-number = account.account-number
```
- Application program이 database에 접근하기 위한 방법
 - Host program 안에 DML을 내장하기 위한 host 언어 문법 확장하는 방법
 - DML과 DDL문을 database에 보내 결과를 얻기 위해, application interface를 제공(e.g. ODBC/JDBC)

데이터베이스 사용자

- 데이터베이스 시스템을 사용하는 사용자들은 시스템과 상호 작용하는 양상에 따라 구분
- 응용프로그램 프로그래머 – DML 호출을 통해 시스템과 상호작용
- 능숙한 사용자 – 질의어를 작성하여 시스템에 대한 요청을 처리
- 특수 사용자 – 전통적인 데이터 처리 방식과는 다른 특수한 데이터베이스 응용을 작성하는 능숙한 사용자
- 일반 사용자 – 미리 만들어진 응용 프로그램들 중 하나를 이용하여 시스템을 사용
 - 예) 웹을 통해 데이터베이스에 접근하는 사람들(은행 직원)

데이터베이스 관리자

- Database administrator: DBA
- 데이터베이스의 모든 권한을 중앙 제어
- 데이터베이스 관리자는 회사의 정보 자원과 수요에 대한 해박한 지식이 있어야 함
- 데이터베이스 관리자가 하는 일:
 - 스키마 정의
 - 저장 구조와 액세스 방법의 정의
 - 스키마 및 물리적 구조의 수정
 - 데이터베이스의 접근 권한 부여
 - 구체적인 무결성 제약조건
 - 요구사항에 대응, 성능 모니터링

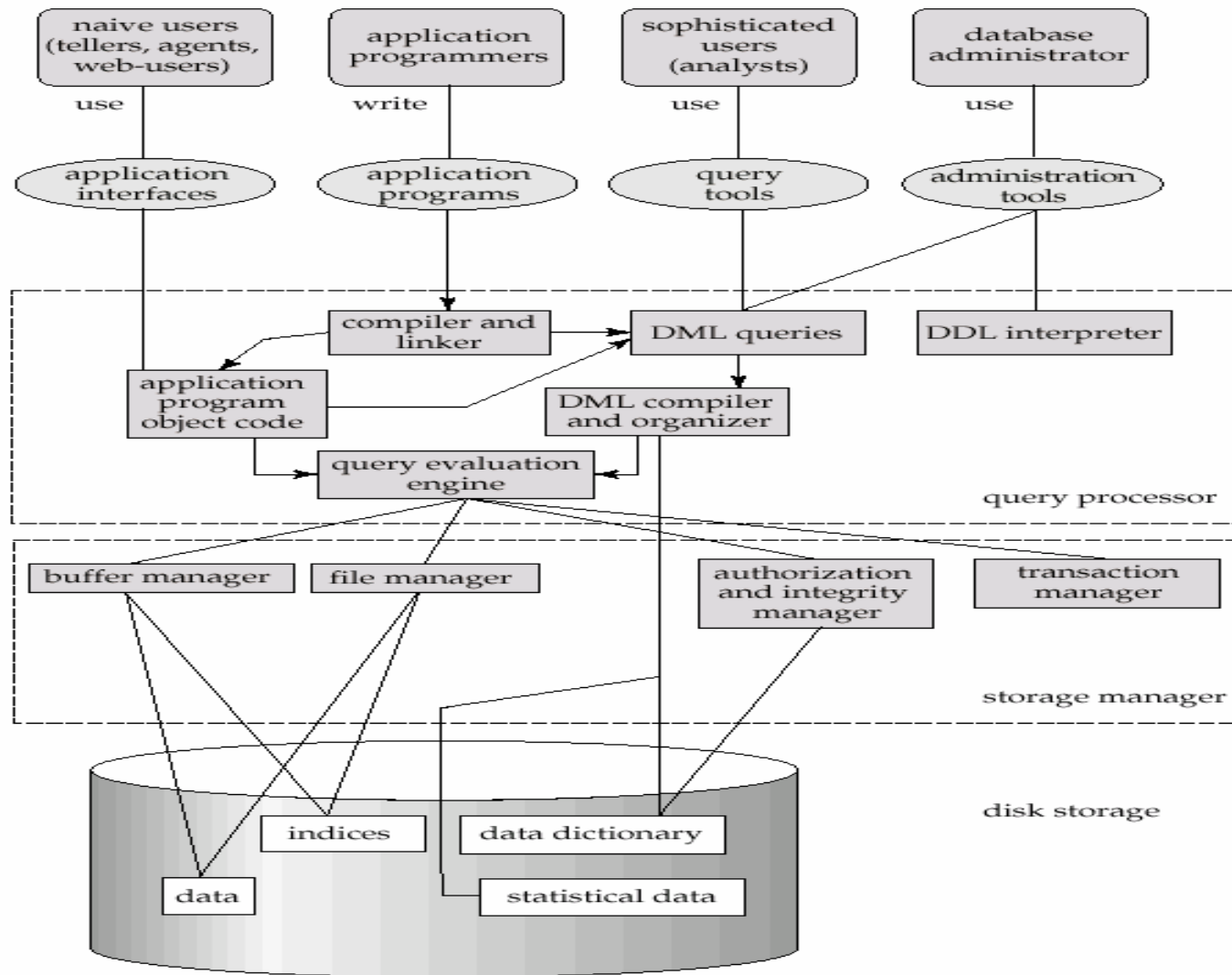
트랜잭션 관리

- 트랜잭션(transaction)
 - 데이터베이스 응용에서 하나의 논리적 기능을 수행하는 연산들의 모임
- 트랜잭션 관리 모듈은 시스템 실패(failure)이나 트랜잭션 실패(전원 문제, OS 다운)가 있을지라도 일관성을 유지하도록 하는 책임을 가짐
- 동시성 제어 관리자(concurrency-control manager)는 데이터베이스의 일관성을 책임지기 위하여, 동시 실행 트랜잭션 간의 상호작용을 제어

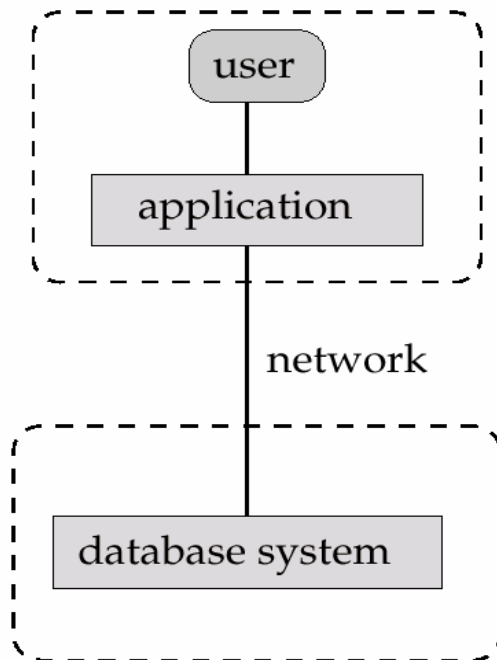
저장 관리

- 저장 관리자는 데이터베이스에 저장된 하위 단계의 데이터와 시스템의 응용 프로그램 및 질의어 사이의 인터페이스를 제공하는 프로그램 모듈
- 저장 관리자의 구성요소:
 - 파일 관리자와의 상호작용
 - 효율적인 저장과 데이터의 복구와 갱신

시스템 구조



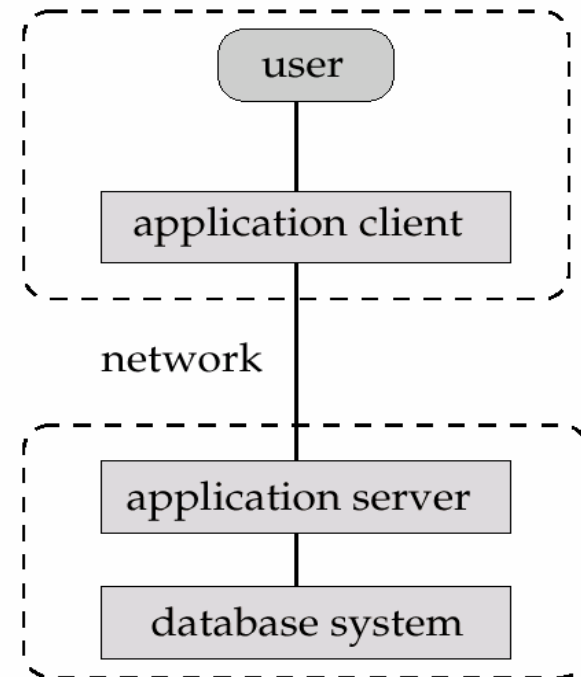
2-계층 구조와 3-계층 구조



a. two-tier architecture

client

server



b. three-tier architecture

데이터베이스 시스템의 역사

- 1950년대와 1960년대 초반
 - 자기 테이프, 천공 카드, 프린터
 - 순차적 접근
- 1960년대 후반부터 1970년대
 - 하드 디스크 - 데이터 처리 과정을 획기적으로 변화시킴
 - 임의 접근
 - 네트워크형 데이터베이스, 계층형 데이터베이스 등장
 - 1970년 Codd가 관계형 모델 발표
- 1980년대
 - IBM - System R → SQL/DS
 - IBM DB2, Oracle, Ingres, DEC Rdb
- 1990년대 초반
 - 데이터 분석 도구, 병렬 데이터베이스
- 1990년대 후반
 - 24시간 가동
 - 웹 인터페이스