1.다음을 간단히 설명하라.

가) logical data independence vs physical data independence

Logical data is data about database, that is, it stores information about how data is managed inside. For example, a table (relation) stored in the database and all its constraints, applied on that relation.

Logical data independence is a kind of mechanism, which liberalizes itself from actual data stored on the disk. If we do some changes on table format, it should not change the data residing on the disk.

All the schemas are logical, and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data.

For example, in case we want to change or upgrade the storage system itself − suppose we want to replace hard-disks with SSD − it should not have any impact on the logical data or schemas.

논리적 데이터는 데이터베이스에 대한 데이터입니다. 즉, 데이터가 내부에서 관리되는 방식에 대한 정보를 저장합니다. 예를 들어, 데이터베이스에 저장된 테이블 (관계) 및 해당 관계에 적용되는 모든 제약 조건.
논리적 데이터 독립성은 디스크에 저장된 실제 데이터로부터 자유 로워지는 메커니즘의 일종입니다. 테이블 형식을 약간 변경하면 디스크에있는 데이터가 변경되지 않아야합니다.
모든 스키마는 논리적이며 실제 데이터는 디스크에 비트 형식으로 저장됩니다. 물리적 데이터 독립성은 스키마 또는 논리적 데이터에 영향을주지 않으면 서 물리적 데이터를 변경할 수있는 힘입니다.
예를 들어, 스토리지 시스템 자체를 변경하거나 업그레이드하려는 경우 - 하드 디스크를 SSD로 교체하려는 경우 논리 데이터 나 스키마에 영향을주지 않아야합니다.

나) ODBC

Open DataBase Connectivity
ODBC는 호출 수준 인터페이스로, 응용 프로그램은 이 인터페이스를 통해 ODBC 드라이버가 있는 모든 데이터베이스의 데이터에 액세스할 수 있습니다. ODBC를 사용하면 최종 사용자가 적절한 ODBC 드라이버가 있는 모든 데이터베이스에 액세스할 수 있는 데이터베이스 응용 프로그램을 만들 수 있습니다. ODBC에서 제공하는 API를 이용하면 소스 DBMS(데이터베이스 관리 시스템)에 종속되지 않는 응용 프로그램을 만들 수 있습니다.
ODBC는 Microsoft WOSA(Windows Open Services Architecture)의 데이터베이스 부분으로, 이 인터페이스를 이용하면 각 플랫폼에 대해 Windows 기반 데스크톱 응용 프로그램을 다시 작성하지 않아도 응용 프로그램을 다중 컴퓨팅 환경에 연결할 수 있습니다.

다) total participation of ERD

- Total Participation − Each entity is involved in the relationship. Total participation is represented by double lines.
- Partial participation − Not all entities are involved in the relationship. Partial participation is represented by single lines.

라) Foreign key

외래키란 테이블 내의 열 중 다른 테이블의 기본키를 참조하는 열을 외래키라 한다.
해당 컬럼 값은 참조되는 테이블의 컬럼 값 중 하나와 일치하거나 null 값을 가짐

마) horizontal and vertical scaling of resources

Horizontal Scaling - also referred to as "scale-out" is basically the addition of more machines or setting up a cluster or a distributed environment for your software system. This usually requires a load-balancer program which is a middle-ware component in the standard 3 tier client-server architectural model.
Vertical Scaling - also referred to as "scale-up" approach is an attempt to increase the capacity of a single machine : By adding more processing power By adding more storage More memory etc Summary:
수평 스케일링 (Horizontal Scaling) - 스케일 아웃 (Scale-Out)은 기본적으로 더 많은 기계를 추가하거나 소프트웨어 시스템을위한 클러스터 또는 분산 환경을 설정하는 것입니다.
여기에는 보통 표준 3 계층 클라이언트 - 서버 아키텍처 모델의 미들웨어 구성 요소 인로드 밸런서 프로그램이 필요합니다.
수직 스케일링 (Scale-up) 접근법은 단일 머신의 용량을 증가시키려는 시도입니다 : 더 많은 프로세싱 파워 추가 스토리지를 추가함으로써 더 많은 메모리 등

2. 관계형 시스템의 최적기는 어떠한 사항을 고려하여 사용자의 요청을 구현할 전략을 작성하는가를 다음 예를 사용하여 설명하라

Let us suppose the user request

- RESULT := (EMP Where EMP# = 'E5') [SALARY]

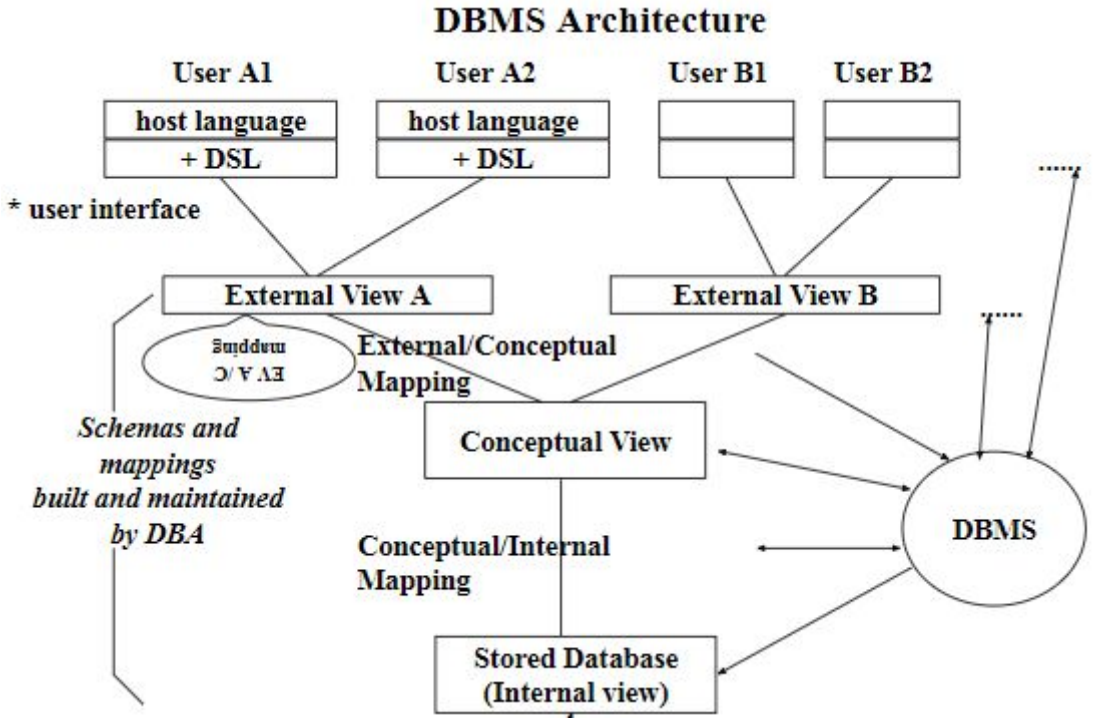
- two ways of performing the necessary data access
  - By doing a physical sequential scan of(the stored version of) table EMP until the required record is found
  - If there is an index on(stored version of) the EMP# column of that table, then by using that index and thus going directly to the E5 data.
    - EMP# is primary key
    - Most systems require an index on the primary key

3. 데이터베이스에서 데이터에 대한 3단계 뷰를 유지하는 근본적인 목적을 설명하고, 이것에 기초하여 특정 레코드의 검색과정을 단계별로 기술하라.

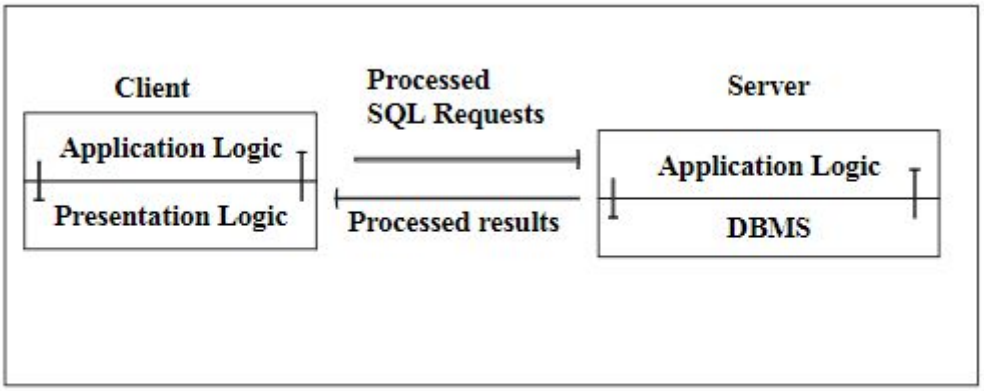
data independence - conceptual schema를 효율성과 응용 데이타 요구사항에 관계없이 상대적으로 안정된 것으로 유지
사용자가 보는 view와 데이타가 저장되는 방법에서의 flexibility와 adaptability

1. External Schema
   : 한 응용에 관심이 있는 data object view (user interface)
   : data language (DDL, DML)
   : individual user view
2. Conceptual Schema
   : 한 조직의 DB를 전체입장에서 보는 것
   : security and integrity 확인
   : community user view
3. Internal Schema
   : physical storage structure
   : define the various type of stored records
   : storage view

4. Mapping
   1) conceptual/internal mapping
      ⇒ ***If the structure of the stored database is changed, then the conceptual/internal mapping must be changed accordingly***
   2) External/conceptual mapping
      ⇒



4. 클라이언트-데이터베이스 서버 구조를 설명하고 이것이 갖는 장점을 기술하시오. 일반적인 two-tier 클라이언트/서버 구조에서 발생하는 문제점을 기술하고, three-tier구조에서 이러한 문제들의 처리과정을 설명하라. 특히 웹 환경에 적합한 구조를 기술하고 그 이유를 설명하라

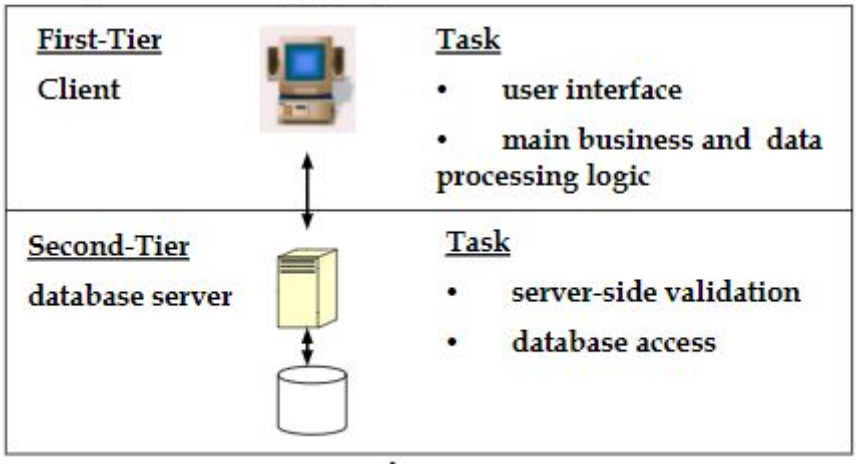


Distribution of processing in client/server model

- Benefits of Client/Server Computing
  - ❑ Dollar Savings
  - ❑ Increased Productivity
  - ❑ Flexibility and Scalability
  - ❑ Resource Utilization
  - ❑ Centralized Control
  - ❑ Open Systems —> standard

Two –tier Client-Server architecture

- **Client**
  - **Responsible for the representation of data to the user**
- **Server**
  - **Responsible for supplying data services to the client**



Three-tier Architecture

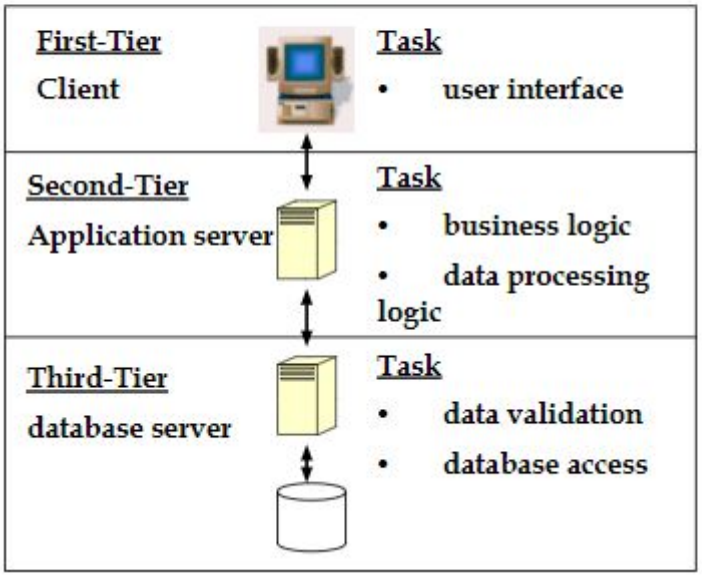
- Mid-1990's (The problem of enterprise scalability)
  - As applications became more complex and potentially could be deployed to hundreds or thousands of users, the client side presented two problems that prevented true scalability.
    - Fat client
    - A significant client-side administration overhead
  - Three layers
    1. The user interface layer (client)
    2. The business logic and data processing layer (application server)
    3. A DBMS (database server)

two-tier 문제점

- 사용자수 증가에 따른 네트웍 트래픽의 병목현상이 발생하여 성능이 현저하게 저하
- 클라이언트에 분산된 애플리케이션 관리가 어렵다
- 데이타베이스 벤더에 종속되어 확장성이 적다
- 애플리케이션 로직이 프리젠테이션 로직에 포함되어 있어 재사용이 어렵다.

Sogang
Univ.

Three-tier Architecture



- Advantages
  - "thin" client
  - App. maintenance is centralized (S/W distribution)
  - Modularity
  - Load balancing

클라이언트와 서버 사이에 애플리케이션 서버를 두어 로직을 분리하고 미들웨어로서 클라이언트와 데이타베이스를 연결하여 부하를 분산하고자 함

- 애플리케이션의 분산으로 performance 향상
- 서버 기종이나 데이터 베이스에 무관하여 확장성이 용이

- 애플리케이션 집중관리로 유지보수와 재사용이 용이
- 장애에 대한 복구가 용이

적용 업무

-LAN 및 WAN이 혼재된 이질적 분산환경 : 보험, 주문, 재고

- 온라인 트랜젝션이 많은 업무

5. 다음의 내용을 반영한 마포회사의 데이터베이스를 위한 ERD을 작성하시오.

회사 데이터베이 스는 고용인(employee), 부서(department)와 프로젝트(project)에 대한 내용만을 포함한다. (20)

가) 회사는 여러 부서(department)들로 구성되어 있고, 각 부서는 유일한 이름(name)과 번호(number), 그리고 위치를 가지며 부서를 관리하는 한 고용인이 있다. 이 고용인이 부서를 관리하기 시작했을 때(start date)에 대한 정보를 유지한다. 한 부서는 여러 개의 위치를 가질 수 있다.

나) 한 부서는 여러 개의 프로젝트(project)를 제어(control)하며, 각각의 프로젝트는 유일한 이름, 번 호와 위치(location)를 가진다.

다) 각 고용인(employee)의 이름, 주민등록번호, 주소, 봉급(salary), 성별과 생일에 대한 정보를 유지 한다. 고용인은 하나의 부서에 속하지만 같은 부서에 의해서 제어되지 않는 여러 개의 프로젝트에서 일할 수 있다. 각 고용인이 각 프로젝트에서 일한 주당 근무시간을 유지한다. 각 고용인의 직속 상 사(direct supervisor) 정보를 유지한다.

라) 보험 목적을 위하여 각 고용인의 부양가족(dependent)에 대한 정보를 유지한다. 각 부양가족의 이 름, 성별, 출생일과 고용인과의 관계(relationship)를 유지한다.

테이블 이름 department
VARCHAR DEPARTMENT_NAME
INT DEPARTMENT_NUMBER
VARCHAR LOCATION
VARCHAR EMPLOYEE_NAME
DATE START_DATE

테이블 이름 project
VARCHAR PROJECT_NAME
INT PROJECT_NUMBER
VARCHAR LOCATION

테이블 이름 employee
VARCHAR EMPLOYEE_NAME
VARCHAR EMPLOYEE_IDENTITY_NUMBER
VARCHAR EMPLOYEE_ADDR
VARCHAR EMPLOYEE_SALARY
VARCHAR EMPLOYEE_SEX_TYPE
DATE EMPLOYEE_BIRTH_DAY
VARCHAR EMPLOYEE_DEPARTMENT_NAME
VARCHAR EMPLOYEE_PROJECT_NAME
TIME EMPLOYEE_PROJECT_TIME
VARCHAR EMPLOYEE_DIRECT_SUPERVISOR
VARCHAR EMPLOYEE_DEPENDENT_NAME
VARCHAR EMPLOYEE_DEPENDENT_SEX
DATE EMPLOYEE_DEPENDENT_BIRTH_DAY
VARCHAR EMPLOYEE_DEPENDENT_RELATIONSHIP

6. 다음의 릴레이셔날 스킴을 사용하여 답하라. 밑줄은 기본 키 속성을 의미한다.
S(S#, SNAME, STATUS, CITY)
P(P#, PNAME, COLOR, WEIGHT, CITY)
SP(S#, P#, QTY)
다음의 뷰 정의를 가정한다.
CREATE VIEW GOOD_SUPPLIER AS
SELECT S#,STATUS,CITY FROM S WHERE STATUS > 15;
다음의 질의를 고려하자.
SELECT S#,STATUS FROM GOOD_SUPPLIER WHERE CITY = "SEOUL" ;
이 질의가 원하는 것이 무엇인가? 이 질의를 DBMS가 처리하는 과정을 설명하라.(10)

정답
다음은 S# , SNAME , STATUS , CITY를 포함하는 S테이블을 이용하여 S 테이블의 STATUS 상태가 15 이상인 값을 참조하여 S#,STATUS 값을 보여주는 GOOD_SUPPLIER 라는 이름을 가진 뷰 를 생성했다.

SELECT S#,STATUS FROM GOOD_SUPPLIER WHERE CITY = "SEOUL" ;
질의문을 실행하면

질의가 원하는 것
GOOD_SUPPLIER 뷰에서 S# 과 STATUS 값을 가져 온다 단 이 값들은 CITY 의 값이 = SEOUL 의 값이 여야만 한다.

DBMS 질의가 처리 되는순서,

1. GOOD_SUPPLIER 뷰를 참조한다.
2. S# 과 STATUS 값을 가져온다.
3. 뷰의 값에서 S 테이블을 참조하여 CITY = "SEOUL" 값만을 골라낸다.

7. ERD에 적합한 릴레이션 스킴들을 만드는 경우를 고려한다.

가) ERD 상의 composite attribute와 multivalued attribute는 어떻게 처리하는 것이 적당한가?

- "다치 애트리뷰트"(multi-valued attribute)는 이중선(double line)으로 둘러싸인 타원으로표현된다. 적어도 한 개체의 인스턴스에 대해서 한 개 이상의 값을 가질 수 있는 애트리뷰트이다.

there are many distinct values entered for it in the same column of the table. This is called a **multivalued attribute**. The problem with doing it is that it is now difficult (but possible) to search the table for any particular hobby that a person might have, and it is impossible to create a query that will individually list the hobbies that are shown in the table.

- "합성 애트리뷰트"(composite attribute)는 두 개 이상의 애트리뷰트를 포함할 수 있는 애트리뷰트를 말한다. 자기 자신도 포함할 수도 있다. 예를 들면 주소는 번지, 도시 등등의 애트리뷰트들로 이루어진 합성 애트리뷰트라 할 수 있다.

An attribute is considered composite if it comprises two or more other attributes.

Consider an attribute such as name that comprises first and last names. For example, suppose an employee's name is John McKenzie. The first name is John and the last name is McKenzie. It is easy to appreciate that one application may only want the last name, another may display the first name followed by the last name, and yet another application may

display the last name, a comma, and then the first name.

나) ERD상의 Specialization 을 어떻게 처리하는 것인 적당한가?

하나의 개체집합을 여러 개의 하위 개체집합으로 분류

- Top-down design process;
- These subgroupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Depicted by a *triangle* component labeled ISA (E.g. *customer* "is a" *person*).
- **Attribute inheritance** – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.
- 이러한 하위 그룹은 상위 수준 엔터티 집합에 적용되지 않는 특성을 갖거나 관계에 참여하는 하위 수준 엔터티 집합이됩니다.
- ISA라고 표시된 삼각형 구성 요소 (예 : 고객은 "사람")로 표시됩니다.
- 속성 상속 - 하위 레벨 엔티티 세트는 링크 된 상위 레벨 엔티티 세트의 모든 속성 및 관계 참여를 상속합니다.