Chapter 02

> 복습 문제

2.1

데이터 모델: 데이터베이스의 구조를 명시하기 위해 사용할 수 있는 개념들의 집합.

데이터베이스 스키마: 데이터베이스의 기술. 데이터베이스에서 자료의 구조, 자료의 표현 방법, 자료 간의 관계를 형식 언어로 정의한 구조이다. 스키마 다이어그램으로 표시함

데이터베이스 상태: 어떤 특정 시점에 데이터베이스에 들어 있는 데이터. 스냅샷이라고도 한다.

내부 스키마: 스키마 3층 구조 중 하나. 내부 단계가 가지는 스키마로 데이터가 디스크에 어떻게 저장되어 있고, 어떻게 접근해야 하는지와 같은 데이터베이스의 물리적 저장 구조를 기술한다.

개념 스키마: 스키마 3층 구조 중 하나. 개념 단계가 가지는 스키마로 모든 사용자를 위한 전체 데이터베이스의 구조를 기술한다. 내부 스키마의 물리적 구조를 몰라도 어떤 관계인지, 어떤 데이터 타입인지, 엔티티는 무엇인지, 제약조건이나 사용자 연산은 무엇인지를 기술한다.

외부 스키마: 스키마 3층 구조 중 하나. 외부 단계가 가지는 스키마로 뷰 단계라고도 한다. 내부와 개념 스키마의 정보를 가지고 다양한 view를 사용자에게 제공한다. 사용자 그룹에게 필요한 데이터베이스의 부분을 기술하고 나머지는 은폐한다.

데이터 독립성: 고수준의 스키마를 변경할 필요 없이 데이터베이스 시스템의 어떤 단계에서 스키마를 변경할 수 있는 능력. 두 레벨(외부와 개념, 개념과 내부) 중에 더 낮은 레벨(개념 또는 내부 단계)의 스키마가 수정되었다면 Ma

pping만 수정하면 되지, 상위 스키마(외부 또는 개념)를 수정할 필요가 없다. 이는 유지, 보수 비용을 낮출 수 있다.

데이터 정의어: DDL. 스키마를 정의하거나 수정 또는 삭제하기 위해 사용하는 언어. DDL로 정의된 스키마는 데이터 사전(카탈로그)에 저장되고, 삭제나 수정이 발생하면 이 내용이 데이터 사전에 반영된다. ex. CREATE, ALTER DROP.

데이터 조작어: DML. 데이터에 대한 검색, 삽입, 수정, 변경 연산 ex. SELECT, INSERT, UPDATE, DELETE

저장 구조 정의어: SDL. 데이터베이스 시스템의 성능을 세밀하게 조정하기 위해서 물리적인 저장 구조를 정의하는 데 사용하고, 대개의 경우에 데이터베이스 관리자가 사용하므로 일반적으로 별도로 유지. DBMS마다 다른데 보통 sql과 크게 다르지 않다고 보는 것 같다.

데이터 조작어: DCL. 내부적으로 필요한 규칙이나 기법을 정의하기 위해 사용하는 언어 ex. COMMIT, ROLLBACK, REVOKE

뷰 정의어: VDL. 사용자 뷰를 명시하고 개념 스키마 사이의 사상(Mapping)을 나타낸다. 하지만 대부분 SQL이 대신 사용됨. ex. CREATE VIEW

질의어: 고수준 데이터 조작어가 그 자체로 대화식으로 사용되면 질의어라고 한다. 일반적으로 고수준 데이터 조작어에서 검색과 갱신 명령들은 대화식으로 사용될 수 있으므로 질의어의 일부로 본다.

호스트 언어: 고수준이든 저수준이든 범용 프로그래밍 언어 내에 데이터 조작어 명령이 삽입된 경우에 프로그래밍 언어는 호스트 언어라 부르고, 데이터 조작어는 데이터 부속어라고 한다.

데이터베이스 유틸리티: DBMS는 데이터베이스 관리자가 데이터베이스 시스템을 관리하는 것을 도와주는 데이터베이스 유틸리티를 가지고 있다. 적재(텍스트 파일이나 순차 파일과 같은 기존의 데이터 파일들을 데이터 베이스에 적재하기 위해 적재 유틸리티를 사용, 보통 DBMS 이전할 때 사용), 백업(데

이터베이스 백업 사본, 데이터베이스 복구), 파일 재조직(파일 구조를 재조직하여 성능 향상을 위함), 성능 모니터링(데이터베이스 사용 통계. 이를 통해 파일 재조직 여부, 인덱스 생성 여부 등을 결정)

카탈로그: 데이터 사전이라고도 함. 데이터베이스에 저장되는 데이터에 관한 정보(메타데이터 ex. 스키마와 제약조건, 설계 결정, 사용 표준, 응용 프로그램 설명, 사용자 정보 등)를 저장. 이것도 하나의 데이터베이스이다.

클라이언트/서버 아키텍처: 2층 구조로 이메일, 프린터와 같은 특정 기능을 갖는 서버와 이를 이용할 수 있는 클라이언트 컴퓨터로 구성한다. 클라이언트에 없는 별도의 기능을 접근하려고 할 때 해당하는 기능을 가진 서버와 연결한다.

3층 아키텍처: 클라이언트와 서버 사이에 중간 단계 즉, 웹서버를 둔다. 이 서버는 응용프로그램을 수행하고, 데이터베이스 서버에 저장된 데이터에 접근하는 데 사용되는 비즈니스 규칙(프로시저 또는 제약 조건)들을 저장한다. 클라이언트 요청이 서버에 직접 닿지 않아 보안을 향상 시킨다. 웹서버는 클라이언트의 요청을 받아 처리하고 데이터베이스 질의와 명령들을 데이터베이스 서버로 보낸다. 데이터베이스 서버에서 처리가 되면 그 데이터를 다시 클라이언트로 보내는 역할을 한다.

n층 아키텍처: 사용자와 저장된 데이터 사이를 좀 더 세밀하게 구성요소들로 나누어 n층 아키텍처를 만들 수 있다. 보통 4-5층. 각 층을 독립적으로 이용할 수 있는 장점이 있다.

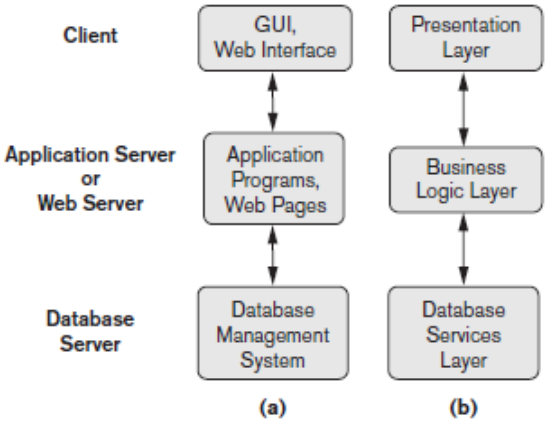


Figure 2.7
Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.

논리적인 3층 클라이언트/서버 아키텍처 <데이터베이스 시스템> p.48

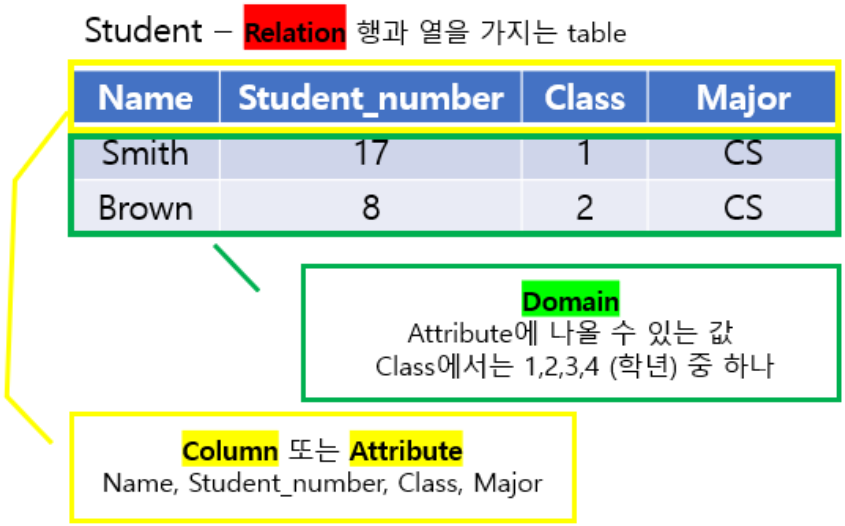
2.2 **체크 필요**

데이터 모델은 크게 4가지로 구분할 수 있다.

데이터 모델 분류 기준	내용
DBMS 기반 기준	현재 상용 DBMS에서 가장 많이 사용되고 잇는 주요 데이터 모델은 관계 데이터 모델. 관계 데이터 모델에 기반한 시스템을 SQL 시스템이라고 한다. 객체 데이터 모델은 일부 상용 시스템에서 구현되었으나 널리 사용되지는 않는다. 키-값 저장 시스템과 NOSQL 시스템과 같은 빅 데이터 시스템은 문서, 그래프, 컬럼, 키-값 등 다양한 데이터 모델을 사용한다. 객체-관계 데이터 모델은 객체과 관계 모델 통합. 꾸준히 발전 중 XML 모델은 실험적이고 트리 구조 데이터 모델
시스템이 허용하는 사용자의 수 기준	단일 사용자 시스템은 한 번에 한 사용자만을 허용. 대부분 개인용 컴퓨터에서 사용 다수 사용자 시스템이 대부분. Multi-User.
데이터베이스가 분산되어 있는 사이트의 수	데이터가 한 컴퓨터 사이트에 저장되어 있다면 중앙집중식. 다수 사용자가 동시에 사용할 수는 있지만 데이터베이스 자체가 한 사이트에만 저장되어 있다. 분산 DBMS는 네트워크로 연결된 여러 컴퓨터 사이트에

	실제 데이터베이스와 DBMS 소프트웨어가 분산되어 있다. (고가용성) 분산 DBMS는 동질과 이질로 나눌 수 있는데 모든 사이트에서 같은 DBMS를 사용하는지 상이한 DBMS를 사용하는지에 따라 나눈다. 이질 분산 DBMS는 클라이언트/서버 아키텍처에서 중간 층 소프트웨어 개발로 느슨하게 결합하고 어느 정도의 지역 자치성을 갖을 수 있다.
비용	오프 소스처럼 무료인 경우와 비용을 지불하는 DBMS로 나눌 수 있다.
기타	저장 파일의 접근 경로의 유형에 따라 나눌 수 있고, 범용 또는 특수 목적용 등으로도 구분할 수 있다.

관계 모델: 3가지 용어(관계relation, 속성attribute, 영역domain)를 먼저 알자. 관계는 행과 열을 가지는 테이블이다. 테이블의 열이 속성이다. 영역은 속성 값이 가질 수 있는 값을 의미한다. 테이블 기반으로 엔티티(Entity)를 테이블과 컬럼의 형태로 데이터를 저장한다. 주로 외래키를 이용하여 테이블 간의 연관관계를 이용해서 정보를 구한다. 예를 들어 개인정보라고 하면 이름, 전화번호, 주소, 주민번호 등 여러 가지가 모여야 개인정보라는 실제 엔티티를 만들 수 있다. 테이블의 행의 순서는 상관없고, 중복된 열이 올 수 없다. 각각의 열은 하나의 속성에 하나의 값만을 가질 수 있다. 나중에 정규화 과정에 대해 이야기하겠지만 정규화 과정이 필요하다.



DBMS 관계 모델 - 관계relation, 속성attribute, 영역domain

객체 모델: 데이터베이스와 응용 프로그래밍의 간격을 줄이는 모델로 데이터베이스를 응용 프로그램과 같은 형태의 시스템으로 사용한다. 이렇게 하는 이유는 데이터베이스의 테이블의 행과 응용 프로그램의 객체 사이의 정보를 변환할 때 오버헤드를 피하고, 객체 프로그래밍의 중요 개념인 캡슐화, 다형성 등을 데이터베이스에 그대로 적용할 수 있다.

객체-관계 모델: 객체 관계 다이어그램(Entity-Relationship Diagram, ERD)이나 객체 관계 맵핑(Object-Relational Mapping, ORM)과 같은 모델과 이들을 직접적으로 지원하지 않는 관계형 데이터베이스를 연결한다. 관계형 데이터베이스와 객체지향 모델링을 하는 자바와 같은 프로그래밍 언어를 지원하기 위함이다. 소프트웨어 개발자들이 그들 스스로 만든 타입과 메서드를 통합시킬 수 있다. 객체 지향 언어의 타입과 데이터베이스의 타입을 일치시키는 맵핑이 주목적이다.

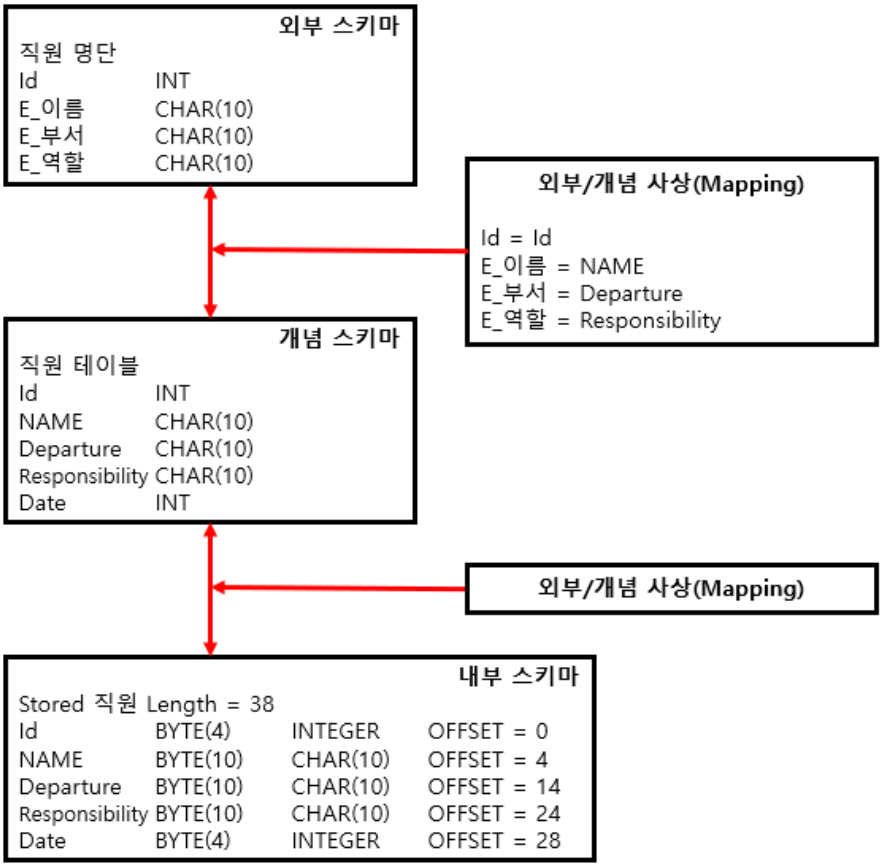
XML 모델: 트리 구조를 가진 데이터 모델로 데이터와 문서 중심인 xml 문서를 이용해서 정보관리, 데이터 교환 및 통합 등을 목적으로 xml 데이터를 저장, 질의, 검색, 조작한다. 자세한 것은 나중에 chapter 12 할 때.

2.3

데이터베이스 스키마: 데이터베이스의 구조와 제약 조건에 관한 전반적인 명세를 기술한 메타데이터의 집합으로 데이터 사전 또는 카탈로그에 저장된다. 데이터베이스 설계 과정에서 명시하며 자주 변경되는 않고, 대부분의 데이터 모델은 도식적으로 스키마를 나타내는 표기법(스키마 다이어그램)을 가지고 있다. 스키마는 개체Entity, 속성Attribute, 관계Relationship으로 구성된다.

스키마의 종류	내용
외부 스키마	사용자의 입장에서 본 데이터베이스 구조. 사용자마다 서로 다른 데이터베이스 스키마를 가진다.
개념 스키마	전체 데이터베이스의 구조와 제약조건, 엔티티, 데이터 타입, 관계, 사용자 연산 등 명세 유지

	한 개의 스키마가 존재하고 여러 사용자가 공유한다.
내부 스키마	물리적인 저장구조와 인텍스와 같은 접근 경로 등이 기술된 스키마 (내부레코드의 형식, 인덱스 유무, 저장 데이터 항목의 표현 방법)



데이터베이스 스키마와 사상(mapping)

데이터베이스 상태: 특정 시점의 데이터베이스의 내용을 의미. 데이터베이스의 갱신 연산이 수행될 때마다 다른 데이터베이스 상태를 얻기 때문에 시간에 따라 계속 달라진다. 데이터베이스의 어커런스(occurance) 또는 인스턴스(instance)들의 현재 집합이다. 주어진 데이터베이스 상태에서 각 스키마 구조물은 자신의 인스턴스들의 집합을 갖는다.

Id	E_이름	E_부서	E_역할
001	이덕화	KBS	배우
002	이경규	MBC	예능
003	김준현	KBS	예능
...			
037	이태곤	SBS	배우
038	이수근	KBS	예능

데이터베이스 상태. length가 38이라서 38개의 데이터가 입력되어 있다고 생각함

2.4 **체크 필요**

3층 스키마 아키텍처는 클라이언트와 데이터베이스 서버 사이에 중간 단계가 있는 아키텍처이다. 중간 층은 응용 서버 또는 웹 서버라고 하는데 이 서버는 응용 프로그램들을 수행하고, 데이터베이스 서버에 저장된 데이터에 접근하는 데 사용되는 프로시저 또는 제약 조건과 같은 비즈니스 규칙들을 저장하여 중간 역할을 수행한다. 데이터베이스의 보안을 향상 시킨다. 외부 스키마인 사용자 인터페이스와 웹 브라우저에서 요청을 받아 중간 서버는 처리하고, 데이터베이스 질의와 명령들을 데이터베이스 서버로 보낸다. 그다음에 데이터베이스 서버에서 처리된 데이터를 클라이언트로 보내는 통로 역할을 한다. 쉽게 말해 외부 스키마는 인터페이스, 개념 스키마는 응용 규칙, 내부 스키마는 데이터 접근의 역할을 한다.

각 스키마간의 연결을 위해 사상(Mapping)이 필요하다. (위 직원 명단 사진 참조) DBMS의 카탈로그에 단계 사이에서 요구와 데이터를 사상시키는 방법에 관한 정보를 포함한다. DBMS는 카탈로그에서 사상 정보를 참조하여 사상을 수행하는 소프트웨어를 추가로 사용할 수 있다.

서로 다른 정의어는 컴파일러를 통해 이해할 수 있는 형태로 변환되는 것 같다. 컴파일러란 특정 프로그래밍 언어로 쓰여 있는 무서를 다른 프로그래밍 언어로 옮기는 일종의 번역, 통역과 같은 역할을 한다. 예를 들어 사용자가 작성한 데이터 언어가 수행될 수 있도록 컴파일러가 이를 변환시킨다. (s.41 그림 참조)

2.5

논리적 데이터 독립성: 외부 스키마와 개념 스키마 사이의 데이터 독립을 의미한다. 개념 스키마에서 스키마를 수정해도 외부 스키마나 사용자의 응용 프로그램에 영향을 주지 않는다. ex. 테이블 1과 2로 만든 뷰가 있다. 개념 스키마에서 테이블 3에 포함된 행과 열을 수정해도 뷰에 영향을 주지 않는다. 단, 테이블 1을 변경했다면 영향을 준다.

물리적 데이터 독립성: 개념 스키마와 내부 스키마 사이의 데이터 독립을 의미한다. 내부 스키마에서 스키마를 수정해도 개념 스키마에 영향을 주지 않는다. ex. 데이터 저장 방식을 수정하거나 인덱스를 수정했다고 개념 스키마에 영향을 주지 않는다.

일반적으로 물리적 데이터 독립성은 디스크에서의 데이터의 정확한 위치, 저장할 때 사용되는 암호화, 압축, 레코드들을 쪼개거나 합치는 등의 물리적으로 상세한 사항을 사용자에게 은닉하는 것이 대부분이다. 따라서 변경을 해도 대부분 독립적으로 영향을 주지 않는다. 하지만 논리적 데이터 독립성은 응용 프로그램에 영향을 주지 않으면서 데이터베이스 전체적인 구조와 제약조건을 변경해야 하기에 더 어렵다.

2.6

절차적 데이터 조작어는 반드시 범용 프로그래밍 언어 내에 삽입해서 사용한다. 데이터베이스로부터 각 레코드 또는 객체 단위로 검색하여 처리한다. 따라서 레코드들의 집합으로부터 각 레코드 별로 검색해서 처리하기 위해 프로그래밍 언어의 반복문 등을 이용할 필요가 있다. 이런 특징 때문에 한 번에 한 레코드 데이터 조작어라고도 한다.

비절차적 데이터 조작어는 복잡한 데이터베이스 연산들을 간결하게 나타내는 데 사용될 수 있다. 많은 DBMS들은 고수준 데이터 조작어를 터미널이나

모니터에서 대화식으로 입력하는 방식과 범용 프로그래밍 언어 내에 삽입하는 방식을 모두 지원 한다. 후자의 경우 프리컴파일러가 프로그램 내에서 데이터 조작어 문장들을 인식하여 DBMS가 처리할 수 있도록 식별할 수 있어야 한다. SQL과 같이 한 데이터 조작어로 여러 개의 레코드를 검색할 수 있기에 한 번에 레코드 집합 데이터 조작어라고 한다.

2.7 s.38

2.8 **체크 필요** With what other computer system software does a DBMS interact?

버퍼 관리 모듈: 버퍼 공간 관리가 성능에 큰 영향을 미치기 때문에 디스크 입출력을 스케줄 하는 자체 버퍼 관리 모듈을 가지고 있다.

저장 데이터 관리자: 디스크에 저장되어 있는 DBMS의 정보(데이터베이스 또는 카탈로그)에 대한 접근을 제어한다.

질의 컴파일러: 질의들을 파싱 하고, 질의 구문과 파일 및 데이터 원소들이 정확한지 입증하고, 내부 형태로 컴파일러 한다.

질의 최적화기: 연산들을 재배치하고, 연산들의 순서를 바꾸고, 중복이 존재하면 제거하고, 효율적인 검색 알고리즘 수행 시 선택한다.

프리 컴파일러: 호스트 프로그래밍 언어로 작성된 응용 프로그램에서 데이터 조작어 명령들을 추출한다. 추출된 명령들은 데이터베이스 접근을 위한 목적 코드로 컴파일하기 위해 데이터 조작어 컴파일러로 보내고, 프로그램에서 데이터 조작어 명령을 제외한 나머지 부분은 호스트 언어 컴파일러로 보낸다.

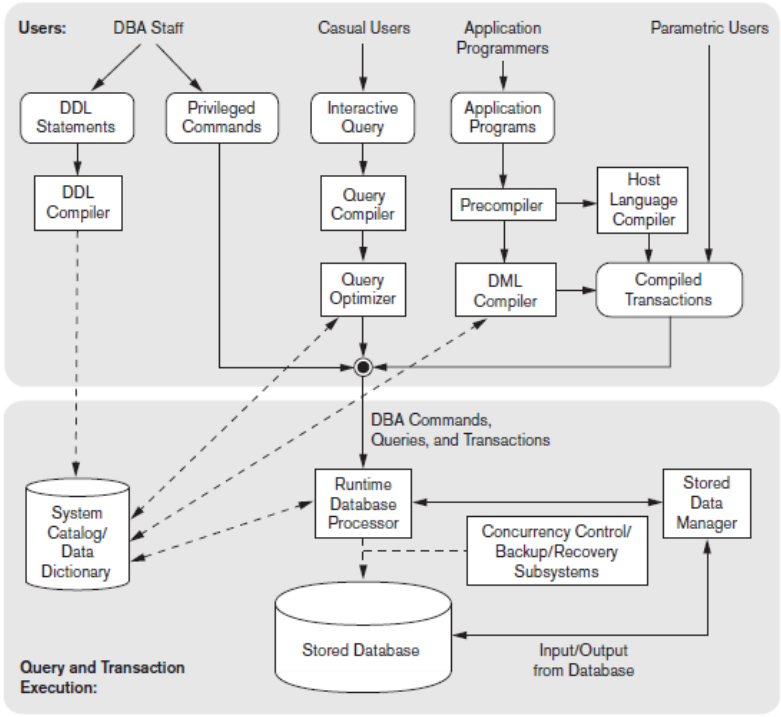


Figure 2.3
Component modules of a DBMS and their interactions.

DBMS를 구성하는 전형적인 모듈들과 이들의 상호 관계

런타임 데이터베이스 프로세서: 시스템 카탈로그와 함께 동작하며 시스템 사전의 통계 정보를 갱신하는 런타임 데이터베이스 프로세서는 특권 명령, 실행 가능한 질의 계획, 런타임시 매개 변수를 받는 미리 작성된 트랜잭션 등을 수행한다. 저장 데이터 관리자(Stored Data Manager)와 함께 동작하기도 하는데 저장 데이터 관리자는 데스크와 주기억 장치 간의 저수준 입출력 연산들을 수행하기 위해서 운영체제의 기본적인 서비스들을 사용한다.

2.9

2층 아키텍처는 클라이언트와 서버를 논리적으로 구분하였다. 서버는 질의와 트랙잭션 기능을 제공한다. 표준 API를 이용하여 필요한 소프트웨어가 클라이언트와 서버 컴퓨터 모두 설치되어 있다면 클라이언트 쪽 프로그램이 DBMS를 호출할 수 있도록 한다. 클라이언트 프로그램은 실제로 여러 RDBMS와 연결하여 ODBC(Open Database Connectivity) API를 통해서 질의와 트랜잭션 요청을 보낼 수 있다. 이런 요청은 서버에서 처리된다. 질의의 결과는

다시 클라이언트 프로그램으로 보내지고, 클라이언트 프로그램은 이 결과를 필요에 따라 좀 더 처리하고 디스플레이할 수 있다. 단순하고 기존의 시스템과의 호환성이 장점이다.

3층 아키텍처는 2.4

2.10 s.42 - s.43

적재: 텍스트 파일이나 순차 파일과 같은 기존의 데이터 파일들을 데이터베이스에 적재하기 위해 적재 유틸리티를 사용한다. migration에 주로 이용.

백업: data recovery

파일 재조직: 성능 향상을 위함. 데이터베이스의 파일들의 구조를 다른 파일 구조로 재조직하고 새로운 접근 경로를 생성

성능 모니터링: 데이터베이스의 사용을 모니터링해서 사용 통계를 데이터베이스 관리자에게 제공. 데이터베이스 관리자는 이런 통계에 근거해서 성능을 향상시키기 위해 파일을 재조직할 것인가 또는 인덱스를 추가하거나 삭제할 것인가 등을 결정한다.

기타: 파일의 정렬, 데이터 압축, 사용자 접근 모니터링, 네트워크와의 인터페이스 등

2.11

n은 보통 4 또는 5. 네트워크를 통해서 프로그램과 데이터를 분산시키는 외에 n층 응용에서는 한 개의 층이 적절한 프로세서나 운영체제에서 동작할 수 있다. 이에 따라 각 층을 독립적으로 다룰 수 있다. 데이터 암호화와 복호화. 하지만 네트워크 보안 문제는 아직 중요한 주제