1 데이터베이스 사용자 분류

01 데이터베이스 사용자 분류

1 개요

- 주소록과 같은 적은 양의 개인정보 데이터베이스는 한 사람의 데이터베이스를 정의, 구축, 관리, 조작할 수 있고 공유되지도 않음
- 그러나 대규모의 데이터베이스에서는 데이터베이스를 정의, 구축, 관리, 조작하는데 많은 사용자들이 관련되어 있음

01 데이터베이스 사용자 분류

2 데이터베이스를 설계, 사용, 관리하는 사용자들

데이터베이스 관리자(Database administrator, DBA)

- 데이터베이스 시스템의 관리를 책임진 사람

(※ 비교 : 시스템 관리자(System Administrator, SA)

데이터베이스 설계자(Database designer)

- 데이터베이스의 설계를 책임진 사람

01 데이터베이스 사용자 분류

2 데이터베이스를 설계, 사용, 관리하는 사용자들

최종 사용자(End users)

- 데이터베이스에 대하여 질의, 갱신, 보고서 작성 등을 담당하는 사람

캐주얼 사용자 (Casual end users)	▪ 비 정기적인 데이터베이스 사용자
초보 사용자 (Parametric or naive users)	▪ 미리 일정한 용도로 작성된 프로그램을 사용하는 사람들 (예, 은행 점원이나 여행사 예약 담당원 등)
전문 사용자 (Sophisticated end users)	▪ 복잡한 응용을 개발하며 DBMS의 기능을 충분히 사용하는 전문가

01 데이터베이스 사용자 분류

2 데이터베이스를 설계, 사용, 관리하는 사용자들

시스템 분석가/응용 프로그래머

- 초보 사용자를 위하여 잘 정의된 기능의 응용을 설계하고 구현하는 사람

01 데이터베이스 사용자 분류

3 DBMS와 시스템 환경을 설계하고 개발하는 사용자들

DBMS의 설계자 및 구현자

- DBMS 소프트웨어 자체를 설계하고 구현하는 업무를 담당하는 사용자들
- DBMS는 매우 복잡한 소프트웨어 시스템으로서 카탈로그 구현, 질의어 처리, 사용자 인터페이스 처리, 데이터 접근과 버퍼링, 동시성 제어, 회복, 보안 등 다양한 모듈로 구성됨
- DBMS는 운영체제나 컴파일러와 같은 시스템 소프트웨어와 연동해서 동작해야 함

01 데이터베이스 사용자 분류

3 DBMS와 시스템 환경을 설계하고 개발하는 사용자들

도구 개발자

- 데이터베이스를 사용하는 데에 필요한 도구들(데이터베이스 설계 및 구축 도구, 성능 도구, 인터페이스 등)을 설계하고 구현하는 사람들

운영 및 유지 보수 요원

- 데이터베이스 시스템을 운영하는 데에 필요한 하드웨어 및 소프트웨어의 운영 및 유지 보수 담당 요원들

2 DBMS의 장점

02 DBMS의 장점

1 개요

- DBMS의 사용에서 얻는 장점과 좋은 DBMS가 가져야 하는 기능들을 설명함
  (※ 이러한 기능들은 1주차 1차시에서 언급한 특징에 추가되는 내용임)

- 데이터베이스 관리자는 대용량의 다수 사용자용 데이터베이스를 설계, 관리, 사용하는데 관련된 다양한 목적을 달성하기 위하여 이러한 특성들을 잘 이용해야 함

02 DBMS의 장점

2 중복성의 제어

- 파일 처리를 이용한 기존의 소프트웨어 개발에서는 모든 사용자 그룹이 데이터 처리 응용을 다루기 위하여 그들 자신의 파일들을 유지함

예시) 대학에서 학적과와 회계과 직원그룹이 있다고 할 때

- 학적과에서는 과목과 성적 등에 관한 정보를 유지함
- 회계과에서는 학생의 등록과 그와 관련된 고지서 정보를 관리함
- 또 다른 부서에서도 이들 정보가 필요하면 **자신의 파일들에 정보를 중복 저장**하게 됨

02 DBMS의 장점

2 중복성의 제어

- 중복성(Redundancy)의 초래하는 문제들
  - 논리적으로 한번의 변경이지만 데이터가 중복된 횟수만큼 반복해서 변경해야 함
  - 디스크 저장 공간과 메모리의 낭비를 초래
  - 동일한 데이터를 포함하는 파일들에서 데이터의 불일치(Inconsistency) 발생 가능

→ 각 논리적 데이터는 데이터베이스 내에 오직 한 번만 저장되는 것이 가장 이상적임

→ 그러나 **질의 성능**을 **개선**하기 위해 **제어된 중복성**이 유용한 경우도 있음

02 DBMS의 장점

2 중복성의 제어

제어된 중복성의 예

- GRADE_REPORT 레코드에 접근할 때마다 Student Name과 Course Number를 함께 접근하는 응용이 빈번하다면 Student Name과 Course Number를 GRADE_REPORT 파일에 중복 저장하는 것이 더 나음

(a)

GRADE_REPORT	Student Number	Student Name	Section Identifier	Course Number	Grade
	17	Smith	112	MATH2410	B
	17	Smith	119	CS1310	C
	8	**Brown**	85	MATH2410	A
	8	Brown	92	CS1310	A
	8	Brown	102	CS3320	B
	8	Brown	135	CS3380	A

(b)

GRADE_REPORT	Student Number	Student Name	Section Identifier	Course Number	Grade
	17	**Brown**	112	MATH2410	B

[데이터를 중복하여 저장하는 경우와 데이터 불일치의 예]

02 DBMS의 장점

2 중복성의 제어

제어된 중복성의 예

- 이렇게 하지 않으면 GRADE_REPORT 파일의 레코드에 접근한 다음에 Student Name과 Course Number를 얻기 위하여 STUDENT와 SECTION 파일에 다시 접근해야 하므로 검색 비용이 늘어나게 됨

(a)

GRADE_REPORT	Student Number	Student Name	Section Identifier	Course Number	Grade
	17	Smith	112	MATH2410	B
	17	Smith	119	CS1310	C
	8	Brown	85	MATH2410	A
	8	Brown	92	CS1310	A
	8	Brown	102	CS3320	B
	8	Brown	135	CS3380	A

(b)

GRADE_REPORT	Student Number	Student Name	Section Identifier	Course Number	Grade
	17	Brown	112	MATH2410	B

[데이터를 중복하여 저장하는 경우와 데이터 불일치의 예]

02 DBMS의 장점

2 중복성의 제어

제어된 중복성의 예

- 중복의 허용은 이러한 경우에 유용하지만 데이터 불일치 현상을 초래하지 않도록 제어해야 함

(a)

GRADE_REPORT	Student Number	Student Name	Section Identifier	Course Number	Grade
	17	Smith	112	MATH2410	B
	17	Smith	119	CS1310	C
	8	Brown	85	MATH2410	A
	8	Brown	92	CS1310	A
	8	Brown	102	CS3320	B
	8	Brown	135	CS3380	A

(b)

GRADE_REPORT	Student Number	Student Name	Section Identifier	Course Number	Grade
	17	Brown	112	MATH2410	B

[데이터를 중복하여 저장하는 경우와 데이터 불일치의 예]

02 DBMS의 장점

2 중복성의 제어

제어된 중복성의 예

- 이러한 중복성의 제어는 GRADE_REPORT 파일의 각 레코드에 나타난 Student Number-Student Name의 값이 반드시 STUDENT 파일에 포함된 어떤 레코드의 해당 속성의 값과 일치하도록 함으로써 자동화할 수 있음

(a)

GRADE_REPORT	Student Number	Student Name	Section Identifier	Course Number	Grade
	17	Smith	112	MATH2410	B
	17	Smith	119	CS1310	C
	8	Brown	85	MATH2410	A
	8	Brown	92	CS1310	A
	8	Brown	102	CS3320	B
	8	Brown	135	CS3380	A

(b)

GRADE_REPORT	Student Number	Student Name	Section Identifier	Course Number	Grade
	17	Brown	112	MATH2410	B

[데이터를 중복하여 저장하는 경우와 데이터 불일치의 예]

02 DBMS의 장점

2 중복성의 제어

제어된 중복성의 예

- 유사하게 GRADE_REPORT 파일이 각 레코드에 나타난 Section Identifier-Course Number의 값이 반드시 SECTION 파일에 포함된 어떤 레코드에서 해당 속성의 값과 일치하도록 해야 함

(a)

GRADE_REPORT	Student Number	Student Name	Section Identifier	Course Number	Grade
	17	Smith	112	MATH2410	B
	17	Smith	119	CS1310	C
	8	Brown	85	MATH2410	A
	8	Brown	92	CS1310	A
	8	Brown	102	CS3320	B
	8	Brown	135	CS3380	A

(b)

GRADE_REPORT	Student Number	Student Name	Section Identifier	Course Number	Grade
	17	Brown	112	MATH2410	B

[데이터를 중복하여 저장하는 경우와 데이터 불일치의 예]

02 DBMS의 장점

2 중복성의 제어

제어된 중복성의 예

- 데이터베이스 설계할 때 이러한 값의 일치 여부에 대한 검사 규정을 한 번 설정해 둠으로써 GRADE_REPORT 파일이 변경될 때 마다 DBMS가 자동으로 일치하는 레코드의 존재 여부를 검사함
- 즉 데이터의 중복을 DBMS가 자동으로 제어하게 됨

(a)

GRADE_REPORT	Student Number	Student Name	Section Identifier	Course Number	Grade
	17	Smith	112	MATH2410	B
	17	Smith	119	CS1310	C
	8	**Brown**	85	MATH2410	A
	8	Brown	92	CS1310	A
	8	Brown	102	CS3320	B
	8	Brown	135	CS3380	A

(b)

GRADE_REPORT	Student Number	Student Name	Section Identifier	Course Number	Grade
	17	**Brown**	112	MATH2410	B

[데이터를 중복하여 저장하는 경우와 데이터 불일치의 예]

02 DBMS의 장점

2 중복성의 제어

제어된 중복성의 예

- 그림 (a),(b)는 아래의 STUDENT 파일과 불일치하는 GRADE_REPORT 레코드를 보여줌
- 17번 학생은 Brown이 아니라 Smith임
- 중복성이 제어되지 않으면 이러한 레코드가 GRADE_REPORT에 삽입될 수 있음

(a)

GRADE_REPORT	Student Number	Student Name	Section Identifier	Course Number	Grade
	17	Smith	112	MATH2410	B
	17	Smith	119	CS1310	C
	8	**Brown**	85	MATH2410	A
	8	Brown	92	CS1310	A
	8	Brown	102	CS3320	B
	8	Brown	135	CS3380	A

(b)

GRADE_REPORT	Student Number	Student Name	Section Identifier	Course Number	Grade
	17	**Brown**	112	MATH2410	B

[STUDENT 파일]

STUDENT	Name	Student Number	Class	Major
	Smith	17	1	CS
	Brown	18	2	CS

[데이터를 중복하여 저장하는 경우와 데이터 불일치의 예]

02 DBMS의 장점

3 권한이 없는 접근의 통제

- 다수 사용자가 대용량의 데이터베이스를 공유하는 경우에 대부분의 사용자는 전체 데이터베이스를 볼 필요가 없거나 보아서는 안 됨
- 일반적으로 사용자 또는 사용자그룹은 패스워드로 보호되는 계정을 부여 받아 데이터베이스를 접근할 때 사용
- DBMS는 이를 위하여 보안과 권한(Security and authorization) 서브시스템을 가짐
- DBA는 이 서브시스템을 사용하여 데이터베이스 사용자에게 계정을 부여하며 접근 통제를 명시함

02 DBMS의 장점

3 권한이 없는 접근의 통제

- 그런 다음, DBMS는 데이터가 접근되기 전에 지정된 접근 통제를 시행함으로써 권한이 없는 사용자의 접근을 방지
- 데이터에 대한 접근 통제와 더불어 DBMS 자체의 실행을 제한할 수도 있음

예시)

- 새 계정을 만드는 등의 특권 소프트웨어 패키지는 DBA만 실행할 수 있도록 할 수 있으며, 초보 사용자는 미리 작성된 응용프로그램만 사용하여 데이터베이스를 접근할 수 있도록 제한 할 수 있음

02 DBMS의 장점

4 프로그램 객체를 위한 지속성 기억 공간 제공

- 객체지향 데이터베이스 시스템의 주요 목적
- 프로그래밍 언어는 대개 복잡한 자료구조를 갖음 (ex. C++나 Java의 클래스 정의가 복잡한 자료구조의 예)
- 또한 프로그램의 변수의 값은 프로그램의 수행이 끝나는 순간 사라지며, 이를 지속시키려면 프로그램 내에서 변수의 값을 파일에 저장해야만 함

02 DBMS의 장점

4 프로그램 객체를 위한 지속성 기억 공간 제공

- 그런데 복잡한 자료구조를 갖는 값을 파일에 저장할 때 파일 포맷에 적합하도록 변형하는 것이 중요함
- 역으로 파일로부터 데이터를 검색할 때는 파일 포맷으로 저장된 데이터를 프로그램 변수 구조로 변환하는 것이 필요함
- 객체 지향 DBMS는 프로그래밍 언어와 호환성을 가지며 필요한 포맷의 변환을 자동으로 수행함

02 DBMS의 장점

5 효율적인 질의 처리를 위한 저장 구조 제공

- 데이터베이스 시스템은 질의와 갱신을 효율적으로 수행하는 기능을 제공해야 함
- 일반적으로 데이터베이스는 디스크에 저장되기 때문에 DBMS는 원하는 레코드를 디스크에서 검색하는 시간을 줄이기 위해 특수한 자료구조를 제공해야 함
- 인덱스라고 불리는 보조 파일이 이런 한 목적에 사용됨
- 인덱스는 대체적으로 트리 자료구조나 해시 자료구조에 기반하고 있고, 디스크 검색을 위해 적절하게 변형됨

02 DBMS의 장점

5 백업과 회복 제공

- DBMS는 하드웨어와 소프트웨어의 고장으로부터 복구할 수 있는 기능을 가져야 함
- 트랜잭션 수행 도중에 컴퓨터 시스템이 고장 나면 DBMS는 갱신 트랜잭션이 수행되기 전의 상태로 데이터베이스를 되돌리거나 고장 난 시점으로부터 나머지를 실행하여 갱신 트랜잭션의 결과가 완전히 데이터베이스에 반영되도록 함

02 DBMS의 장점

5 다양한 사용자 인터페이스 제공

- 사용자들은 기술적인 지식의 보유 정도에 따라서 다양한 그룹으로 나누어지므로 DBMS도 이들 각각에 적합한 인터페이스를 제공해야 함
- 캐쥬얼 사용자에게는 질의어를, 응용 프로그래머에게는 프로그래밍 언어 인터페이스를, 초보 사용자에게는 폼이나 그래픽 기반 인터페이스를, 전문적인 사용자에게는 메뉴기반 인터페이스나 커맨드라인 인터페이스를 지원해야 함
- 웹 GUI나 웹 연동 데이터베이스 접근 기능이 점점 일반화되어 가고 있음

02 DBMS의 장점

6 데이터 간 복잡한 관계의 표현

- 데이터베이스에는 다양한 형태로 연관된 데이터가 저장됨
- DBMS는 새로운 관계가 나타날 때 그 관계를 정의하고 연관된 데이터를 쉽고 효율적으로 검색, 변경할 수 있는 기능을 제공하기 위해 데이터 간의 복잡하고 다양한 관련성을 표시할 수 있어야 함

02 DBMS의 장점

7 무결성 제약조건(Integrity constraints, IC)의 시행

- 미니월드를 정확하게 반영하기 위해서 DBMS가 제약조건들을 정의하고 검사하는 기능을 가져야 함
- 제약조건은 실세계의 의미로부터 발생하는 것이며, 데이터베이스 설계자가 설계 시에 무결성 제약조건도 같이 명시하게 됨
- 엔티티 무결성 제약조건, 관계 무결성 제약조건, 키 무결성 제약조건, 참조 무결성 제약조건, 유일성 무결성 제약조건, 업무 규칙(Business rule)무결성 제약조건 등이 있음

02 DBMS의 장점

8 규칙을 사용한 추론한 수행

- 연역 데이터베이스 시스템(Deductive database system)은 데이터베이스에 저장되어 있는 사실로부터 새로운 정보를 추론하는 연역적 규칙을 정의할 수 있음
- 관계 DBMS에서는 테이블을 트리거(Trigger)와 연관시킬 수 있음
- 트리거는 테이블의 갱신에 의해 활성화되는 규칙의 한 형태로 다른 테이블에 추가적인 동작을 수행하거나 메시지를 보내는 등의 일을 수행함
- 능동 데이터베이스 시스템(Active database system)은 어떤 이벤트나 조건이 일어났을 때 자동으로 수행되는 능동규칙을 제공함

02 DBMS의 장점

9 데이터베이스 사용에 함축된 또 다른 의미

표준화된 데이터 관리

- 조직 내 모든 부서에서 표준화된 문서 관리로 업무 효율성 증대
- 데이터 구조 변경에 융통성 부여
- 데이터베이스 내의 자료 구조가 어떠한 이유로 변경되어도 사용자에 대한 영향은 거의 없음

02 DBMS의 장점

9 데이터베이스 사용에 함축된 또 다른 의미

응용 프로그램의 개발 시간 단축

- 응용 프로그램의 상당한 부분을 DBMS가 처리함
- DBMS를 사용할 때의 개발시간은 전통적인 파일 시스템을 사용할 때 보다 1/4 에서 1/6 정도로 적게 걸리는 것으로 추정

02 DBMS의 장점

9 데이터베이스 사용에 함축된 또 다른 의미

융통성

- 요구사항이 변하면 데이터베이스의 구조가 바뀔 수 있음
- 기존의 데이터베이스에 파일을 추가하거나 기존의 파일에 데이터 항목을 추가하는 것이 필요할 수 있음
- 현대의 DBMS들에서는 저장된 데이터나 현재 사용하는 응용 프로그램에 영향을 미치지 않으면서 데이터베이스의 구조를 조금씩 변경하는 가능함

02 DBMS의 장점

9 데이터베이스 사용에 함축된 또 다른 의미

최신 정보의 가용성

- 사용자 중 한 사람의 갱신으로 나머지 사람은 즉시 변경된 값에 접근하는 것이 가능

02 DBMS의 장점

9 데이터베이스 사용에 함축된 또 다른 의미

규모의 경제성(Economics of scale)

- DBMS를 사용하면 조직 내의 데이터와 응용이 통합되어 관리되므로 응용 프로그램들 사이의 중복 뿐만 아니라 서로 다른 프로젝트나 부서의 업무가 불필요하게 중복되는 것을 상당 부분 줄일 수 있음
- 각 부서에서 장비를 따로 구매하고 정보를 별도로 관리하는 것보다, 동일한 예산으로 전체 조직이 더 좋은 처리기, 저장장치, 통신장치 등에 투자할 수 있음
- 전체적인 운영 경비를 절감

3 데이터베이스의 발전 과정

03 데이터베이스의 발전 과정

1 개요

DBMS를 이용한 응용의 역사와 이 응용이 새로운 형태의 데이터베이스 시스템 출현에 어떠한 영향을 미쳤는지 간략하게 살펴봄

03 데이터베이스의 발전 과정

2 데이터베이스를 사용한 응용 개발 시작

- 계층 모델과 네트워크 모델 등이 60년대 중반부터 80년대까지 주류를 이루었음
- 초기의 데이터베이스 응용은 회사, 대학, 병원, 은행 같은 큰 조직에서 사용하는 레코드들을 관리
- 이러한 응용에서는 비슷한 구조의 레코드들을 주로 다룸

03 데이터베이스의 발전 과정

2 데이터베이스를 사용한 응용 개발 시작

- 주된 문제점 중의 하나는 개념적인 관계와 물리적 저장 공간, 그리고 디스크에서 레코드의 위치가 혼동되어 사용되는 것이었음
- 이는 원래의 질의나 트랜잭션에 대해서는 유연성을 제공하지만 새로운 형태의 질의나 트랜잭션에 대해서는 유연성을 제공하지 못함
- 특히 서로 다른 저장 공간 구성을 필요로 하는 새로운 형태의 질의를 효율적으로 처리하는 데 어려움이 많았음

03 데이터베이스의 발전 과정

3 관계형 데이터베이스를 통하여 응용 유연성 확대

- 관계형 모델은 70년대 소개 이후, IBM과 세계 각 대학에서 연구되고 검증되어 왔으며, 80년대에 들어서 상용 DBMS가 등장함
- 원래 데이터의 물리적 저장공간과 그 개념적인 표현을 분리하고 데이터베이스에 수학적인 근거를 제공하기 위해 제안됨
- 프로그래밍 언어의 대안으로 제공되었던 고차원 질의를 소개함
- 고차원 질의를 사용하면 새로운 질의를 훨씬 빠르게 작성할 수 있음

03 데이터베이스의 발전 과정

3 관계형 데이터베이스를 통하여 응용 유연성 확대

- 그 이전 시스템들의 응용과 똑같은 응용을 목표로 하였지만, 새로운 질의를 빨리 개발하고 요구사항이 바뀌었을 때 데이터베이스를 빨리 재구성할 수 있는 유연성을 제공

- 80년대 초기에 소개된 상업적인 관계 데이터베이스 관리 시스템(Relational Database Management System; RDBMS)는 데이터 레코드에 접근하기 위해 물리적 저장 공간 포인터나 레코드 위치 정보를 사용하지 않았기 때문에 상당히 속도가 느림

03 데이터베이스의 발전 과정

3 관계형 데이터베이스를 통하여 응용 유연성 확대

- 새로운 저장 공간, 인덱스 기술, 그리고 더 나은 질의 처리와 최적화 방법이 개발되면서 성능이 더욱 향상됨
- 결국, 관계 데이터베이스는 전통적인 데이터베이스 응용을 위한 데이터베이스 시스템의 주된 형태가 되었음

03 데이터베이스의 발전 과정

4 객체지향 응용과 더욱 복잡한 데이터베이스에 대한 요구

- 80년대 객체지향 프로그래밍 언어의 출현과 복합구조 객체를 저장하고 공유할 필요로 인해 객체 지향 데이터베이스(Object-oriented database; OODB)가 개발됨
- 초기에는 객체지향 데이터베이스가 관계 데이터베이스보다 더 일반적인 자료구조를 제공했기 때문에, 객체지향 데이터베이스는 관계 데이터베이스의 경쟁자로 간주되었음

03 데이터베이스의 발전 과정

4 객체지향 응용과 더욱 복잡한 데이터베이스에 대한 요구

- 객체지향 데이터베이스는 추상형 자료구조, 연산자의 캡슐화, 상속, 다형성, 객체 식별자와 같은 많은 유용한 패러다임을 제공하였음
- 하지만 모델의 복잡성과 초기 표준의 부재로 인해 그 사용은 제한적임
- 현재 객체지향 데이터베이스는 주로 공학 디자인, 멀티미디어 출판, 제조 시스템 등과 같은 특수한 응용에서 주로 사용됨
- 객체지향 데이터베이스가 큰 영향을 줄 것이라는 예상에도 불구하고, 현재 데이터베이스 시장에서의 보급률은 5% 이하임

03 데이터베이스의 발전 과정

5 전자 상거래(E-Commerce)를 위한 웹에서 데이터 교환 필요성 등장

- 월드 와이드 웹이 등장으로 하이퍼텍스트 마크업 언어(Hyper text markup language; HTML)와 같은 웹 출판 언어를 통해 문서를 생성하고, 이 문서를 다른 사용자들이 접근할 수 있도록 웹 서버에 저장할 수 있음
- 90년대에는 전자상거래가 웹의 주요 응용 중의 하나로 출현함
- 실제 전자상거래 웹 페이지에 있는 정보의 일부는 DBMS에 의해 동적으로 추출된 것들임

03 데이터베이스의 발전 과정

5 전자 상거래(E-Commerce)를 위한 웹에서 데이터 교환 필요성 등장

- 웹 상에서 데이터의 교환을 허용하기 위해 다양한 기술이 개발되었고 지금은 XML이 다양한 형태의 데이터베이스와 웹 페이지 사이에서 데이터를 교환하는 표준으로 인식되고 있음

03 데이터베이스의 발전 과정

5 새로운 응용을 위한 데이터베이스 능력 확장

새로 대두되는 응용들

- 고에너지 물리학 또는 인간 유전자 지도와 같은 분야의 과학적 실험으로 얻어지는 대용량의 데이터를 저장하는 과학적 응용
- 스캔한 뉴스 이미지, 개인 사진에서 위성 사진에 이르는 여러 이미지, 그리고 x-레이나 MRI 와 같은 의학적 이미지들을 저장하고 검색하는 응용
- 영화와 같은 비디오, 또는 신문이나 디지털 카메라로부터 얻은 비디오 클립의 저장과 검색 응용

03 데이터베이스의 발전 과정

5 새로운 응용을 위한 데이터베이스 능력 확장

새로 대두되는 응용들

- 특정한 패턴이나 관계를 검색하면서 대량의 데이터를 분석하는 데이터 마이닝 응용
- 특정 위치의 날씨 정보나 지도와 같이 위치 데이터를 저장하는 공간(Spatial) 응용
- 정기적인 경제 데이터와 같은 정보를 저장하는 시계열(Time series) 응용

03 데이터베이스의 발전 과정

5 새로운 응용을 위한 데이터베이스 능력 확장

관계 데이터베이스가 새로운 응용에 적합하지 않은 이유

- 모델링을 위해서는 단순한 관계형 표현보다 더욱 복잡한 자료구조가 필요함
- 기본적인 수치형과 문자형 타입 외에 새로운 데이터 타입이 필요함
- 새로운 데이터 타입을 다루기 위해서는 새로운 연산과 질의어 구조가 필요함
- 새로운 저장 공간과 인덱스 구조가 필요함

03 데이터베이스의 발전 과정

5 새로운 응용을 위한 데이터베이스 능력 확장

이로 인해 DBMS개발자들은 그들의 시스템에
새로운 응용을 지원하기 위한 기능을 추가하게 됨

- 객체지향 데이터베이스를 관계데이터베이스에 통합하는 형태의 범용 기능
- 특정 응용에서 사용될 수 있게 만들어진 부가적인 모듈 형태의 특수 목적의 기능

03 데이터베이스의 발전 과정

5 새로운 응용을 위한 데이터베이스 능력 확장

데이터베이스를 사용하지 않아도 되는 경우

- DBMS를 사용하면 비용이 높아짐
- 높은 초기 투자 비용과 추가적인 하드웨어 필요함
- 데이터의 보안, 동시성 제어, 회복, 무결성 제약조건 등의 기능이 필요하지 않은 응용에서는 오히려 오버헤드가 됨

03 데이터베이스의 발전 과정

5 새로운 응용을 위한 데이터베이스 능력 확장

언제 DBMS가 불필요한가?

- 데이터베이스와 응용이 단순하고 잘 정의되어 있으며, 변경될 가능성이 적을 경우
- DBMS 오버헤드로 인하여 엄격한 실시간 데이터 처리 요구사항을 만족시키기 힘든 경우
- 사용자의 데이터 접근이 필요하지 않은 경우