협성대학교 소프트웨어공학과

데이터베이스 개론

9~13 주차 강의 정리

이시헌

2024-6-13

9 주차 강의 정리 시작.

1. 서론
   A. 교재 8 장과 9 장은 데이터베이스를 설계하는 방법을 설명한다.
   B. 8 장은 [E-R 모델 + 릴레이션 변환 규칙]을 사용한 DB 설계를 다룬다.
   C. 9 장은 정규화를 이용한 DB 설계를 다룬다.

2. E-R 모델과 릴레이션 변환 규칙을 사용한 DB 설계 과정
   A. 설계 절차는 다음과 같다.
      ① 요구사항 분석
      ② 개념적 설계 - E-R 다이어그램 제작.
      ③ 논리적 설계 - E-R 을 릴레이션 스키마로 변환. (논리적 스키마.)
      ④ 물리적 설계 - 물리적 스키마 설계.
      ⑤ 구현 - 실제 SQL 전송기능 작성, DBMS 에서 DB 생성 등….
   B. 자세한 것은 따로 설명하겠다.

3. 요구사항 분석
   = 사용자의 니즈 파악. (일반적인 SW 설계 절차와 거의 동일하다.)
   A. 가장 중요한 단계이다. 요구 사항 분석이 잘못되면 사용자가 원치 않는 쓸모 없는 DB 가 개발되어, 고치거나 다시 개발하는 등의 수고가 뒤따를 수 있다.
   B. 주요 목적: DB 의 용도 파악.
   C. 결과물: 요구사항명세서.
   D. 분석 절차.
      ① 사용자의 범위를 결정.
         -〉 불필요한 요구사항 수집 차단.
      ② 해당 사용자가 조직에서 수행하는 업무 분석.
         -〉 어떤 데이터를 필요로 하는가? 그 데이터에 어떠한 처리가 필요한가? 등.
      ③ 분석한 요구 사항을 요구 사항 명세서로 문서화.

4. 개념적 설계

= 요구분석 단계의 결과물을 개념적 데이터 모델을 통해 표현하는 단계.

A. 결과물: 개념적 스키마(E-R 다이어그램).

B. E-R 다이어그램: 데이터베이스에 저장해야 하는 요소를 추출하고, 요소 간의 관계를 파악하여 다이어그램으로 표현한 것.

C. 개념적 모델링: 사용자의 요구사항을 개념적 데이터 모델로 변환하는 작업. (모델링의 결과를 E-R 다이어그램으로 표현할 수 있다.)

D. 개념적 스키마(구조): E-R 다이어그램 등의 수단으로 표현된 개념적 설계의 결과물.

E. 과정

① 개체 및 속성 추출

- 저장할 만한 가치가 있는 중요 데이터를 지닌 사람이나 사물 등을 뽑아낸다.
- 일반적으로 문장에서의 명사이며, 광범위한 의미를 지닌 단어를 제외한다.
- 골라낸 단어들을 객체와 속성으로 분리하여 연결한다.

② 관계 추출: 개체 간의 관계를 결정짓는 것.

- 일반적으로 요구사항명세서에서 동사로 표현된다.
- 업무 처리와 관련하여 개체 관의 연관성을 의미 있게 표현한 동사만을 선택한다.
- 관계에 참여하는 객체들 간의 관계 유형 또한 정의한다. (일대다, 다대다, 등….)

③ E-R 다이어그램 작성

- 앞서 두 단계에서 이미 개체에 대한 다이어그램과, 개체간의 관계 다이어그램을 모두 작성했다.
- 작성된 다이어그램들을 하나로 합하면, E-R 다이어그램이 완성된다.

5. 논리적 설계

= 개념적 스키마를 기반으로 논리적 스키마를 설계하는 작업.

= DMBS 에 적합한 논리적 데이터 모델을 이용해서 논리적 스키마를 설계하는 작업.

A. 이 시점에서 어떤 DMBS 를 사용할지 선택한다.

B. 결과물: 릴레이션 스키마.

C. E-R 다이어그램을 릴레이션 스키마로 변환하는 규칙이 있다.
(해당 규칙을 사용하지 않고 변환하기는 상당히 어렵다.)

D. 릴레이션 스키마 변환 규칙 다섯가지.

① 모든 개체는 릴레이션으로 변환한다.

- E-R 다이어그램의 각 개체를 하나의 릴레이션으로 취급한다.
- 개체의 이름 = 릴레이션의 이름.
- 개체의 속성 = 릴레이션의 속성.

② 다대다 관계는 릴레이션으로 변환한다.

- 관계 자체를 릴레이션으로 만든다는 말이다.
- [관계의 주체인 개체]들은 규칙 1 에 따라 변환하고,
  [변환된 개체의 기본키]를 [관계 릴레이션]에 외래키로 포함시킨다.
- 관계의 이름 = 릴레이션 이름.
- 관계의 속성 = 릴레이션 속성. (개체에만 속성이 붙는 것이 아님을 기억하라. 관계에도, 연산을 위한 속성이 부여되어 있을 수 있다.)

③ 일대다 관계는 외래키로 변환한다.

- 일대 다 관계에서 '1'측 개체의 기본키를 'n'측개체에 외래키로 포함시킨다. (릴레이션으로 변환하지 않는다.)
- 약한 개체가 연관된 경우에도 위의 작업은 동일하게 수행한다.
  다만 [n 측 개체의 기본 키]에 [1 측 객체에서 가져온 키(외래키)]를 포함시켜야 한다. (약한 개체는 강한 개체에 의해 존재 여부가 결정되므로 그렇다. 혹은, 강한 개체에서 필수적으로 약한 개체로 접근 가능해야 하므로 기본 키를 묶어서 정의한다고 생각하면 되겠다.)

④ 일대일 관계를 외래키로 표현한다.

- 데이터의 중복을 피하기 위한 세부 규칙이 있다.
- 일반적인 1 대 1 관계에서는 외래키를 서로 주고받는다.
- [일대일 관계에 필수적으로 참여하는 개체]가 있다면,
  [필수개체]가 [선택적 개체의 기본키]를 외래키로 받는다.
- 모든 개체가 필수적으로 참여하는 일대일 관계라면,
  모든 개체를 하나의 릴레이션으로 표현한다.
  ([[모든 개체의 기본키]를 릴레이션의 외래키]로 받고,
  [모든 외래키를 묶어서 기본키]로 설정한다. p.306 참고)

⑤ 다중 값 속성은 릴레이션으로 변환한다.

- E-R 모델에서 다중 값을 가지는 속성은 허용되지 않는다.
  -〉 다중 값 속성을 릴레이션으로 만들 때, [[외래키]로 다중 값 속성이 속했던 개체의 기본키]를 가져온다. (p.307 참고)

⑥ 참고. 속성이 많은 관계는, 유형에 상관 없이 릴레이션으로의 변환을 고려할 수 있다.

E. 위의 변환 규칙을 모두 적용하여 얻어낸 릴레이션을 기반으로, 테이블 명세서를 제작하게 된다.

① 사용할 DBMS 를 기준으로,
[속성 이름]-[데이터타입]-[NULL 허용 여부]-[키 종류]-[제약조건]-… 등을 정의한다.

6. 물리적 설계

A. 릴레이션 스키마의 설계가 완료되면, 물리적 설계를 시작한다.

B. 물리적 설계 절차.

= 하드웨어와 운영체제의 특성을 고려하여, 필요한 인덱스의 구조나 내부 저장 구조, 접근 경로 등에 대한 물리적인 구조를 설계한다.

① DBMS 를 이용해 SQL 문을 작성하고 실행시켜 DB 를 실제로 생성한다.

7. 구현

= DBMS 를 이용해 SQL 문을 작성하고, 이를 실행하여 DB 를 실제로 생성한다.

9 주차 강의 정리 끝. (8 장 정리 끝.)

10~11 주차 강의 정리 시작.

1. 서론

A. 9 장의 내용은 '정규화'이다.

B. 앞서 설명했지만, DB 를 설계하는 대표적 방법 중 하나이다.

2. 정규화

= DB 의 이상 현상(부작용)들을 제거하는 것.

= DB 의 삽입, 삭제, 수정 연산시에 발생하는 부작용들을 제거하면서 DB 를 올바르게 설계해 나가는 작업.

= 이상 현상이 발생하지 않도록, 릴레이션을 분해하는 작업.

= 관련이 없는 함수 종속성은 별개의 릴레이션으로 표현하는 작업.

= '함수적 종속성'을 기준으로 릴레이션을 분해하는 작업.

3. 이상 현상

= 잘못 설계된 DB 에서 발생하는, 일어나서는 안되는 현상.

A. 일반적으로 불필요한 데이터 중복 때문에 발생한다.

B. 세 종류가 있다.

① 삽입 이상: 불필요 데이터 함께 삽입.
(새 데이터를 삽입하기 위해, 불필요한 데이터도 함께 삽입해야 하는 문제 발생.)

② 갱신 이상: 중복된 투플 중 일부만 갱신.
(중복 투플(객체, 행) 중 일부만 변경하여, 데이터가 불일치하게 되는 모순 발생.)

③ 삭제 이상: 꼭 필요한 데이터까지 함께 삭제됨. (연쇄삭제현상)
(투플을 삭제하면 꼭 필요한 데이터가 함께 삭제되는 데이터 손실 발생.)

4. 함수 종속

A. [하나의 릴레이션을 구성하는 속성들]의 부분집합을 X 와 Y 라고 하자.

B. [릴레이션 내의 모든 투플(행)]에서 X 값에 대한 Y 값이 항상 하나이다.

= “X 가 Y 를 함수적으로 결정한다.”

= “Y 가 X 에 함수적으로 종속되어 있다.”

= X → Y

① X: 결정자.

② Y: 종속자.

C. 예시

① 고객아이디→(고객이름, 등급)

고객아이디	고객이름	등급
apple	정소화	gold

② 즉, 아이디가 apple 인 사람은 단 하나이므로, 아이디가 다른 속성들을 종속시키고 있는 것이다.

③ 혹은 다음과 같이 두 줄로 표현 가능.

고객아이디 → 고객이름

고객아이디 → 등급

D. 일반적으로 투플을 유일하게 구분하는 기본키와 후보키는,
릴레이션을 구성하는 다른 모든 속성들을 함수적으로 결정한다.

E. 하지만 기본키나 후보키만 결정자가 될 수 있는 것은 아니다.
-〉 속성 X 값이 같을 때, 이 값과 연관된 모든 속성 Y 가 같은 경우라면, 무엇이든 속성 X 가 될 수 있다.

F. 예시
① 고객아이디→고객이름
{고객아이디, 이벤트번호}→당첨여부
{고객아이디, 이벤트번호}→고객이름

고객아이디	이벤트번호	당첨여부	고객이름
apple	E001	Y	정소화

② 고객 이름은 아이디에 종속된다.
③ 당첨 여부는 아이디와 이벤트 번호 모두에 종속된다.
(어느 하나만으로는 당첨 여부 속성을 결정지을 수 없다.)
④ 고객이름은 아이디 뿐만 아니라, 이벤트 번호에도 종속되어 있다. (고객이름은 기본 키 집단에 종속되어 있다.)

G. 완전 함수 종속
① 이전의 예시에서의 '당첨 여부'가 완전 함수 종속된 종속자이다.
② 릴레이션에서 [속성 집합 Y]가 [속성 집합 X]에 함수적으로 종속되어 있을 때, [속성 집합 X 전체]에 종속된 경우를 말한다. (X 의 일부에 종속된 것이 아닌 경우.)

H. 부분 함수 종속
① 이전의 예시에서의 '고객 이름'이 부분 함수 종속된 종속자이다.
② 릴레이션에서 [속성 집합 Y]가 [속성 집합 X]의 일부에 종속되어 있는 경우이다.

5. 정규형

= 정규화의 강도를 여섯 단계로 나누어 유형화 시킨 것. (지침)

A. 기본 정규형 4 종과, 고급 정규형 2 종으로 나뉜다.

B. 점차 제약조건이 강해질수록, 데이터 중복이 줄어든다.

C. 기본 정규형

① 제 1 정규형

- 모든 속성이 더 분해되지 않는 원자 값 만 갖도록 함.

② 제 2 정규형

- 제 1 정규형 만족
- + 기본키가 아닌 모든 속성이 기본키에 완전 종속된다.

③ 제 3 정규형

- 제 2 정규형 만족.
- + 이행적 함수 종속을 제거.

  (하나의 키가 둘 이상의 속성을 종속 시키는 것을 해소.

  = 하나의 키가 하나의 속성만 종속 시키도록 릴레이션을 분해한다.

  = 기본키가 아닌 모든 속성이 기본키에 이행적 함수 종속이 되지 않게 릴레이션을 분해한다.)

④ 보이스/코드 정규형

- 제 3 정규형 만족.
- 모든 결정자가 후보키를 만족해야 한다.

  (후보키: 유일성과 최소성을 만족하는 속성.

  유일성: 투플들을 구분지을 수 있는 능력.

  최소성: 꼭 필요한 단일 속성일 때 만족됨.)
- 후보키가 아닌 속성을 분리하여, 릴레이션으로 독립시킨다.

D. 고급 정규형

일반적으로 4 정규형까지 진행해야 할 상황은 잘 오지 않는다.

(강의 시간에도 다루지 않음.)

① 제 4 정규형

② 제 5 정규형

-참고. 9 장 끝.

10 장: 회복과 병행 제어

1. 트랜잭션

= 논리적인 작업의 단위.

= 하나의 작업을 처리하기 위한 연산들을 모아 둔 것.

= 작업 수행에 필요한 SQL 문의 모임.

A. 트랜잭션의 특성

① 원자성

- 수행 성공 -> 전부 성공을 보장.
- 수행 실패 -> 전부 실패를 보장.
- 이는, 작업에 실패한 경우 원래 상태로 돌아갈 수 있어야 한다는 뜻이다. (실패 -> 모든 연산 취소 보장.)
  (이때 '모든 연산의 취소'는 '회복 연산'으로 칭한다.)

② 일관성

- DB 내부에 모순이 없어야 한다는 뜻.

③ 격리성(고립성)

- 트랜잭션 완료시까지 다른 트랜잭션들이 접근하지 못하도록 차단.
  ([현재 사용중인 중간 연산 결과]로의 모든 접근을 차단한다는 것.)

④ 지속성

- 트랜잭션 성공 이후 DB 의 상태는 유지되어야 함.
- 이는, 장애 발생시 복구가 가능해야 한다는 뜻이다. (회복 가능.)

⑤ 요약 정리

트랜잭션의 특성	DMBS 의 기능
원자성	회복 기능
일관성	병행 제어 기능
격리성	병행 제어 기능
지속성	회복 기능

B. 트랜잭션 연산의 종류

① commit: 트랜잭션이 성공적으로 수행되었음을 선언. -> 작업 완료.

- commit 연산을 수행해야, DBMS 가 작업이 완료되었다고 인지.
- 즉, 실제 작업 완료 -> commit 연산 실행 -> 트랜잭션 완료.

② rollback: 트랜잭션 수행이 실패했음을 선언. -> 작업 취소.

- 트랜잭션 수행 중간에 장애 발생시, rollback 연산을 수행해야 한다.
- 즉, 장애 발생(실패 상태) -> rollback 연산 실행 -> 모든 연산 취소(철회 상태)

C. 트랜잭션의 상태 관련 용어 정리

① 활동 상태: 트랜잭션을 수행중인 상태.

② 부분 완료 상태: 트랜잭션의 마지막 연산이 실행된 직후의 상태.

③ 완료 상태: commit 연산을 실행한 상태.

④ 실패 상태: 장애 발생으로 인해 트랜잭션의 수행이 중단된 상태.

⑤ 철회 상태: 트랜잭션 수행 실패로 rollback 연산을 실행한 상태.

2. 장애

= 시스템이 제대로 동작하지 않는 상태.

A. 다양한 유형이 있다.

① 트랜잭션 장애

- 트랜잭션 수행 중 오류가 발생하여 정상적으로 수행을 계속할 수 없는 상태.
- 논리 오류, 리소스 과다 요구 등의 문제로 발생.

② 시스템 장애

- 하드웨어 결함으로 정상적으로 수행을 계속할 수 없는 상태.
- 하드웨어 이상으로 메인 메모리의 정보가 손실되는 등의 문제로 발생.

③ 미디어 장애

- 디스크 장치의 결함으로 저장된 DB 가 손상된 상태.
- 디스크 헤드의 고장이나 손상으로 발생.

3. 저장장치의 종류

A. 휘발성 저장장치 -> 장애 발생시 데이터 손실.

B. 비휘발성 저장장치 -> 장애 발생시 데이터 손실 없음. (하지만 손실 될 수도 있음.)

C. 안정 저장장치 -> 장애에 영향 X. (회복이 가능하다)

① 비휘발성 저장장치로 복사본 여럿을 만들어 관리하는 방법.

4. 데이터 이동 연산.

A. 데이터 위치: 응용 SW - 메인 메모리 - 디스크

B. 응용 SW 와 메인 메모리 사이의 연산 (SW 기준)

① read(X): 메인 메모리 버퍼 블록에 있는 X 를, 프로그램의 변수로 읽어오는 연산.

② write(X): 프로그램의 변수 값을, 메인 메모리 버퍼 블록에 있는 X 에 기록하는 연산.

C. 메인 메모리와 디스크 사이의 연산. (메모리 기준)

① input(X): 디스크 블록에 저장되어 있는 데이터 X 를 메인 메모리 버퍼 블록으로 읽어오는 연산.

② output(X): 메인 메모리 버퍼 블록에 있는 데이터 X 를 디스크 블록으로 이동시키는 연산.

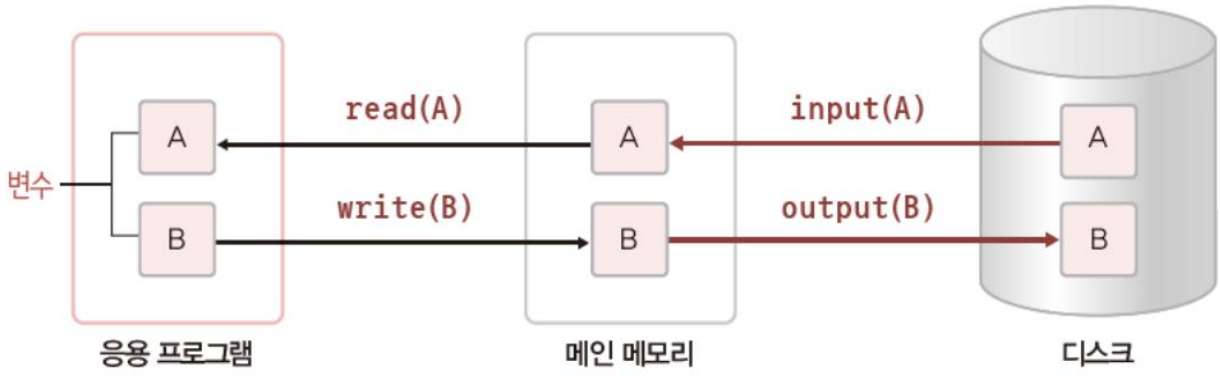

그림 10-18 응용 프로그램이 실행한 트랜잭션의 수행을 위해 필요한 데이터 이동 연산

D.

E. 트랜잭션은 데이터 처리를 위해 데이터를 디스크에서 메인 메모리로 이동시킨다. (input( ))

F. 메인 메모리로 이동시킨 데이터를, 응용 프로그램에서 read( )로 가져올 수 있다.

5. 회복 기법

= 장애 발생시, DB 를 [장애 발생 이전의 일관된 상태]로 복구하는 기법.

A. DBMS 의 회복 관리자가 담당한다.

B. 회복의 핵심 원리는 데이터 중복이다. 미리 복사해 두었던 데이터를 이용하여 원래의 상태로 복원하는 것이다.

C. 데이터 복사본 기록 방식

① dump(덤프): DB 전체를 다른 저장 장치에 주기적으로 복사하는 방법.

② log(로그): DB 에 변경 연산이 실행될 때 마다, 데이터를 변경하기 이전 값과 변경한 이후의 값을 별도로 이 파일에 기록하는 방법.

D. 회복 연산의 종류

① redo(재실행)

- 가장 최근에 저장한 DB 복사본을 가져온다.
- 복사본이 만들어진 이후의 모든 변경 연산을 재실행 한다.
- 전반적인 손상 발생시에 redo 사용.

② undo(취소)

- 로그를 이용해 실행된 모든 변경 연산을 취소한다.
- 변경 중이었거나, 이미 변경된 내용만 신뢰성을 잃은 경우에 사용.

6. 로그 레코드의 종류

A. $\langle T_1$, start$\rangle$ <- 트랜잭션 T_1가 시작했음을 기록.

B. $\langle T_1$, X, old_value, new_value$\rangle$ <- X 를 old_value 에서 new_value 로 변경했음을 기록.

C. $\langle T_1$, commit$\rangle$ <- T_1가 완료되었음을 기록.

D. $\langle T_1$, abort$\rangle$ <- T_1가 철회되었음을 기록.

7. DB 회복 기법

A. 로그 회복 기법

(트랜잭션 연산 결과를 DB 에 반영하는 시점으로 분류 가능하다.)

① 즉시 갱신 회복 기법

= 장애 발생시 로그에 기록된 내용을 참조하여 undo 또는 redo 수행.

(연산 결과를 DB 에 즉시 반영했기에, 로그를 살펴보아야 복구 가능.)

- 트랜잭션 완료 이전 장애 발생 -> undo 수행. (모두 취소)
- 트랜잭션 완료 이후 장애 발생 -> redo 수행. (모두 재실행)

② 지연 갱신 회복 기법

(반영 시점 설명: 트랜잭션이 수행되는 동안에는 데이터 변경 연산의 결과를 DB 에 즉시 반영하지 않고 로그 파일에만 기록 -> 트랜잭션이 부분 완료될 때 마다 로그에 기록된 내용을 이용해 DB 에 한번에 반영한다.)

- 트랜잭션 완료 이전 장애 발생 -> 로그 내용을 무시하고 버림.
  (반영 시점이, 완료 이후이기 때문에, 현재 작업물을 전부 버리면, undo 와 유사한 결과 도출됨.)
- 트랜잭션 완료 이후 장애 발생 -> redo 연산 실행. (모두 재실행)

B. 검사 시점 회복 기법

= 일정 시간 간격으로 검사 시점(check point)을 만들어 두고,

장애 발생시 [가장 최근 검사 시점 이후의 트랜잭션들]에 대해 회복 작업 수행.

① 로그 회복 기법은 많은 연산이 요구된다.

② redo 연산을 수행할 필요가 없는 트랜잭션에도 redo 를 수행해야 하는 일이 발생하기도 함.

③ 이러한 비효율성을 해소하기 위한 좋은 방법이 '검사 시점 회복 기법'이다.

C. 미디어 회복 기법

= 디스크 장애에 대비한 회복 기법.

① 전체 DB 의 내용을 일정 주기마다 다른 안전한 저장 장치에 복사해 둔다. (덤프 저장)

② 디스크 장애 발생

-> 가장 최근 복사한 덤프를 이용해, 장애 발생 이전의 DB 상태로 복구.

③ 덤프를 저장하는 작업에 상당한 비용이 든다는 단점이 있음. (cpu 낭비, 복사 시점의 트랜잭션 수행 불가 등….)

8. 병행 제어

A. 병행 수행: 여러 사용자가 DB 를 동시 공유할 수 있도록, 여러 개의 트랜잭션을 동시에 수행하는 것.

B. 병행 제어: 병행 수행이 가능하도록 트랜잭션 수행을 제어하는 것.

C. 병행 수행에서 발생할 수 있는 문제점들

① 갱신 분실: 일부 변경연산 결과를 덮어쓰는 것으로, 연산이 무효화 됨.

② 모순성: 일관성 없는 상태의 DB 에서 데이터를 가져와 연산할 수 있음.

③ 연쇄 복귀: 장애 발생시, [[장애를 일으킨 트랜잭션이 변경한 데이터]를 사용한 모든 연산]을 rollback 하게 되는 것.

D. 트랜잭션 스케줄

= 트랜잭션에 포함된 연산들을 수행하는 순서.

(인터리빙: 여러 트랜잭션이 번갈아서 연산하는 것.)

① 직렬: 인터리빙 X, 트랜잭션별로 연산들을 순차 처리.

- 항상 모순 없이 정확하다는 장점이 있다.

② 비직렬: 인터리빙 O. 여러 트랜잭션이 번갈아 연산함.

- 앞서 제시한 3 가지 문제가 모두 발생할 수 있다.
- 대책 없이 인터리빙 방식 연산처리를 하면 안된다.

③ 직렬 가능: 직렬 스케줄과 동일한 결과를 생성하는 비직렬 스케줄.

- 직렬 가능성을 보장하는 '병행제어법'이 필요하다!

9. 병행 제어 기법

A. 트랜잭션의 직렬 가능성을 보장하기 위한 규약이 있다.

B. 대표적인 규약으로, '로킹 기법'이 있다.

10. 기본 로킹 규약

= 한 트랜잭션이 먼저 접근한 데이터에 대해, 연산 종료시까지 다른 트랜잭션의 접근을 차단한다. (상호 배제한다.)

A. lock 연산을, read 혹은 write 연산 실행 전에 요청해야 한다.

① lock 연산: 독점권 요청 연산.

② unlock 연산: 독점권 반환 연산.

B. 로킹의 단위가 큰 경우. (투플, 릴레이션 등의 큰 개체에 독점권 요청)

① 병행성 낮다.

② 제어 쉽다.

C. 로킹의 단위가 작은 경우. (속성에 가깝다)

① 병행성 높다.

② 제어 어렵다.

D. lock 연산의 종류

① 공용 lock

- 모두에게 read 허용 + write 불가.
- 여러 트랜잭션이 공용 lock 을 개체에 중복으로 걸 수 있다.

② 전용 lock

- lock 요청한 트랜잭션만 read, write 가능.

11. 2 단계 로킹 규약

A. 기본 로킹 규약의 문제를 해결하기 위해, 새로운 규약을 추가한 것이다.

B. 모든 트랜잭션이 2 단계 로킹 규약을 준수하면, 직렬 가능성이 보장된다.

C. 2 단계 로킹 규약

① 확장 단계

- 트랜잭션이 lock 만 실행 가능, unlock 은 실행할 수 없는 단계.
- 트랜잭션이 처음 수행되면 확장 단계로 진입한다.

② 축소 단계

- 트랜잭션이 unlock 만 실행 가능, lock 은 실행할 수 없는 단계.
- 확장 단계에서 unlock 연산을 실행하는 순간, 축소 단계로 변환된다.
- 일단 축소 단계로 진입하였다면, 모든 lock 연산에 대한 독점권 반환(unlock)을 수행해야만 한다.

D. 모든 lock 연산이, 최소의 unlock 연산 이전에 실행되어야 한다는 뜻이다.

12. 교착 상태(dead lock)

= 트랜잭션이 서로가 독점하고 있는 데이터에 대하여, unlock 이 실행되기를 기다리면서 수행이 중단된 상태.

A. 적극적인 예방이 필요하다.

B. 빠른 탐지 및 해소가 필요하다.

11 주차 강의 끝. + 교재 10 장 정리 끝.

13 주차 강의 정리 시작(12 주차 휴강) + 교재 11 장 정리 시작.

1. 보안
   A. 물리적 환경에 대한 보안 <- 자연재해 등…. 튼튼한 건물 지어라.
   B. 운영 관리를 통한 보안 <- 권한이 있는 사용자로부터 보호….
      ① 올바른 제약조건 설정과, 적절한 직원 교육으로 달성.
   C. 권한 관리를 통한 보안 <- 권한이 없는 사용자로부터 보호….
      ① 가장 중요! 일반적으로 보안 관리는 권한 관리를 칭함.

2. 권한 관리의 개념.
   A. 사용자별, 그룹별 접근 범위를 설정한다.
   B. 수행 가능한 작업 범위를 제약한다.
   C. 권한이 있어야만 작업이 가능하도록 한다.

3. 권한 부여
   A. GRANT **권한** ON **객체** TO **사용자** [WITH GRANT OPTION]
      ① 권한
         - DDL(데이터 정의어. create table, create view….)
           <- DDL 사용시 객체 지정 필요 없음.
         - 테이블 명령어: insert, delete, update, select, references….
      ② 객체: 테이블 이름이다.
      ③ 사용자: 사용자 이름이다.
      ④ WITH GRANT OPTION
         - 해당 문구를 작성하면, 사용자가 자신이 부여 받은 권한을 다른 사용자에게도 부여할 수 있게 된다.
         - 가급적 쓰지 마라.
   B. 예시
      ① GRANT SELECT ON 고객 TO Hong;
         = 고객테이블에 대한 검색 권한을 사용자 Hong 에게 부여하라.
      ② GRANT CREATE VIEW TO Shin;
         = 뷰를 생성할 수 있는 권한을 사용자 Shin 에게 부여하라.
4. 권한 취소

A. REVOKE 권한 ON FROM 사용자 CASCADE | RESTRICT

① 권한

- DDL(create table….) <- DDL 사용시, 객체 지정 필요 없음.
- 테이블 명령어(select….)

② 객체: 테이블 이름

③ 사용자: 사용자 이름

④ CASCADE

- 특정 사용자가 제 3 자에게 준 모든 권한도 함께 취소.
- '연쇄적 권한 취소'라고도 함.
- 제 3 자가 또 다른 제 3 자에게 준 권한들도 모두 뺏는다.
  (N 차 계승된 모든 권한 박탈.)

⑤ RESTRICT

- 특정 사용자의 권한만 박탈한다.

B. 이와 같은 권한 취소 명령을 내리기 위해서는,
누가 무슨 권한을 갖고 있는지 '권한 목록'을 만들어 관리해야 한다.
(권한 목록 <- DB 관리자가 기록, 관리해야 함.)

C. 예시

① REVOKE SELECT ON 고객 FROM Hong CASCADE;
Hong 에게 부여한 고객 테이블에 대한 검색 권한을 취소하면서,
Hong 이 부여한 동일 권한도 모두 취소한다.

② REVOKE CREATE TABLE FROM Hong;
Hong 에게 부여한 테이블 생성 권한 취소.

5. Role

= 특정 사용자에게 부여할 권한을, 이름을 붙여 미리 지정해둔 것.

A. CREATE ROLE 롤 이름;

B. 이때 [롤 이름]에 권한을 부여하거나 취소할 수 있고,
특정 사용자에게 [롤 이름]을 부여할 수 있다.

C. GRANT 권한 ON 객체 TO 롤 이름;
-> REVOKE 롤 이름 FROM 사용자;
-> GRANT 롤 이름 TO 사용자;

D. 역할을 제거할 수도 있다. 역할을 제거하면, 해당 역할을 부여 받은 사용자들도, 해당 권한을 더 이상 지니지 않게 된다.

① DROP ROLE 롤 이름;

② DROP ROLE role_1;

E. 예시

① GRANT role_1 TO Hong;

11 장 끝.

12 장: 데이터베이스 응용 기술

1. 서론

A. 11 장까지 데이터베이스 이론 학습이 끝났다.

B. 12 장은 그것으로 무엇을 할 수 있는지 보여준다.

2. 발전 과정

A. 관계형 DB -> 객체 DB -> 객체관계 DB -> 응용 기술들.

B. 응용 기술 목록

① 분산 DB

② 멀티미디어 DB

③ 웹 DB

④ 데이터 웨어하우스

3. 분산 데이터베이스

= 물리적으로 분산되어 있으나, 논리적으로는 하나로 보이는 데이터베이스.

A. 일반적으로 여러 장소에 데이터를 중복으로 저장한다.

B. 위치 투명성: 사용자는 실제 저장 위치를 알 필요가 없다.

C. 중복 투명성: 사용자는 중복을 인지하지 못한다.

D. 장점

① 한 지역에서 문제가 발생해도, 다른 지역에서 작업 재개 가능.
-> 신뢰성과 가용성 향상.

② 동일한 데이터가 저장된 여러 지역에서 병렬 처리 가능.
-> 데이터 처리 성능 향상.

③ 데이터 처리 요청이 여러 지역에 분산됨.
-> 처리 부담 감소.

E. 단점

① 저장 공간 사용량이 크다.

② 데이터를 변경하려면 중복 저장된 모든 데이터를 함께 변경해야 한다.
-> 비용 증가, 문제(불일치) 발생 가능성 높아짐.

4. 단편화

A. 분산 데이터베이스의 단점(저장공간 낭비)을 해소하는 방법.

B. 하나의 릴레이션을 더 작은 조각(단편)으로 나누고, 각 조각을 별개의 릴레이션으로 처리하는 기법.

C. 단편화 투명성: 사용자는 데이터가 단편화 된 것을 인식할 수 없어야 한다.

D. 단편화의 세 가지 조건.

① 완전성: 전체 릴레이션의 모든 데이터는, 어느 한 조각에는 꼭 속해야 한다.

② 회복성: 단편화된 조각들로부터 원래의 전체 릴레이션을 회복할 수 있어야 한다.

③ 분리성: 전체 릴레이션의 모든 조각을 서로 중복되지 않게 분리해야 한다.

E. 단편화의 종류

① 수평적 단편화

- 릴레이션을 투플(행) 단위로 나누는 것.
- 동일 속성 목록을 지닌 복수의 릴레이션을 만든다고 생각하라.

② 수직적 단편화

- 릴레이션을 속성 단위로 나누는 것.
- 하나의 릴레이션을 세로로 쪼갠다. 그러므로 기본 키를 공유하는 복수의 릴레이션이 생긴다고 생각하라.

③ 혼합 단편화

- 수평적 수직적 단편화를 모두 사용하여 릴레이션을 나눈다.

5. 병행 투명성

= 분산 데이터베이스와 관련된 트랜잭션들이 동시에 수행되더라도, 일관성을 유지하는 것.

6. 장애 투명성

= 특정 지역 시스템에 문제가 발생하더라도, 전체 시스템이 작업을 계속 수행할 수 있는 것.

7. 분산 데이터베이스의 구조

A. 핵심! 계층적인 스키마 구조로 만들어진다는 것이다.

B. 상위 스키마는 하위 스키마 변경에 영향 받지 않아야 한다.

C. 지금까지 설명한 많은 투명성들을 달성하기 위해서는, 계층 구조가 필수적이다.

D. 구조

① 전역 개념 스키마

- 분산 DB 에 저장할 모든 데이터 구조와 제약조건 정의.
- 데이터베이스 안에 존재하는 모든 릴레이션 스키마의 집합.

② 단편화 스키마

- 전역 개념 스키마를 분할하는 방법(단편화)을 정의하는 스키마.
- 단편화: 전역 개념 스키마를 논리적으로 분할한 것.

③ 할당 스키마

- 각 조각 스키마 인스턴스를 물리적으로 저장할 지역을 정의한다. (실제로 하나 이상의 지역에 물리적으로 저장된다.)

④ 지역 스키마 <- 복수 존재.

- 지역별로 저장하고 있는 데이터 구조와 제약조건 정의.

⑤ 데이터베이스 관리 시스템 <- 복수 존재.

- 사실상 독립적인 DBMS 이다. 상위 스키마에서 지역 DBMS 에게 적절한 요청을 하여, 사용자에게 어떻게든 하나의 DBMS 로 보여지게만 하면 되는 것이다.

⑥ 데이터베이스 <- 복수 존재.

- 물리적 저장장치를 포함하는 데이터베이스이다.

8. 분산 데이터베이스 시스템의 장, 단점.

A. 장점

① 신뢰성과 가용성 증대

② 지역 자치성과 효율성 증대

③ 확장성 증대

B. 단점

① 높은 설계 및 구축 비용

② 분산된 여러 지역 모두 관리 필요 -〉 관리 복잡 + 비용 증가

③ 여러 지역의 분산 데이터베이스를 모두 사용하여 사용자 요청 처리 -〉 중앙 집중식 데이터베이스 시스템에 비하여 높은 처리 비용 발생(통신 비용 등…)

9. 멀티미디어 데이터베이스 시스템

= 관계형 DB + 멀티미디어 데이터 저장을 위한 데이터 타입 추가.

A. 현대에 쓰이는 멀티미디어 DB 시스템은, 일반적으로 BLOB 에 대한 관계 연산을 지원한다.

B. BLOB: Binary Large Object = 대형 이진 객체. 〈- 추가된 타입.

C. 멀티미디어 데이터의 특성

① 대용량 〈- 압축 필요성 있음.

② 검색 방법이 복잡

- 설명 기반 검색: 데이터에 대한 설명을 함께 저장해 두었다가, 검색에 활용.
- 내용 기반 검색: 데이터의 실제 내용을 이용하여 검색을 수행한다.

③ 구조가 복잡

- 영상이나 이미지를 떠올려 보라. 어찌나 복잡한지….

10. 웹 데이터베이스

= 웹 서비스와 데이터베이스 시스템을 통합한 것.

A. 적절한 미들웨어가 있어야, 웹 서비스와 데이터베이스가 연결될 수 있다.

① 주로 서버에 미들웨어를 상주시키는 방식을 사용한다.

② 클라이언트 쪽에 미들웨어를 둘 수도 있다.

11. 데이터 웨어하우스

= 데이터 창고.

= 사용자가 원하는 데이터를 빠르게 검색하여 제공 가능한 데이터 창고.

(굉장히 방대한 내용이 있으나, 이번 학기에 다루지 않는다.)

13 주 강의 정리 끝. + 12 장 정리 끝.

14 주차 강의는 '데이터 과학'에 대한 내용이다.

정리를 생략하겠다.

(교재 초반부에서 다루었던 내용 상당수 포함. + 정성적인 데이터 과학에 대한 이해 요구. -> 14 주 필기를 한번 훑어보고 시험에 임하라.)

이상으로 〈데이터베이스 개론 3 판〉(김연희, 한빛아카데미) 8~12 장의 정리가 끝났다.