데이터베이스

장애와 회복

학습목표

- 데이터베이스의 장애 유형과 회복의 기본 원리를 설명할 수 있다.
- 데이터베이스의 회복을 위한 로그의 기본 구조를 설명할 수 있다.

학습내용

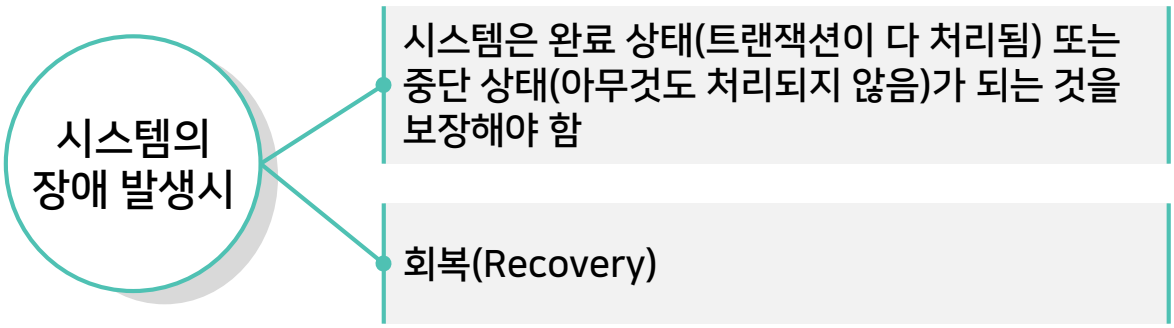
- 회복의 기본 원리
- 로그

회복의 기본 원리

장애

01 장애의 의미

시스템이 정해진 명세대로 작동하지 않는 상태



회복의 기본 원리

장애

02 장애의 유형

전원 오류

트랜잭션 장애

시스템 장애

미디어 장애

전원 오류

트랜잭션 장애

시스템 장애

미디어 장애

- 트랜잭션의 논리적 오류나 내부 조건(입력 데이터의 불량, 시스템 자원의 과다 요구 등)으로 인하여 트랜잭션이 정상적으로 실행할 수 없는 상태

회복의 기본 원리

장애

02 장애의 유형

전원 오류 | 트랜잭션 장애 | **시스템 장애** | 미디어 장애

- 하드웨어 오동작으로 메모리에 있는 정보의 손실이 발생되거나 교착 상태로 인하여 더 이상 시스템이 실행되지 않는 상태

전원 오류 | 트랜잭션 장애 | 시스템 장애 | **미디어 장애**

- 디스크 붕괴 및 고장 등에 의하여 저장 장치의 데이터 일부 및 전체가 손상된 상태

일반적인 장애의 주요 원인

- 사람에 의한 실수

회복의 기본 원리

회복의 원리

01 회복의 의미

장애가 발생되었을 때 데이터베이스를 장애 발생 이전의 일관된 상태로 복원시키는 과정

* 일관된 상태 : 오류가 없는 상태, 즉 데이터베이스 내에 모순이 없는 상태

- 회복의 기본 원리
  - 중복: 데이터를 여러 곳에 복제해 놓으면, 시스템 장애 발생 시 문제 없음

02 회복을 위한 데이터 중복 기법

덤프(Dump) 또는 백업(Backup)	- 데이터 전체나 일부분을 다른 저장 장치로 데이터를 복제 → 주기적으로 데이터베이스 전체를 다른 저장 장치에 복제하는 것
로그(Log)	- 저널(Journal) → 데이터베이스가 변경될 때 마다 변경되는 데이터로 아이템의 옛 값과 새 값을 별도의 화일에 기록해 두는 것

회복의 기본 원리

회복의 원리

03 회복을 위한 조치

재해적 실패(미디어 장애 등)에 대한 회복 | 비재해적 실패에 대한 회복

- 데이터베이스 내용 자체가 손상 된 경우 가장 최근 복제본을 적재 시킨 뒤, 해당 복제본 이후에 일어난 변경만을 로그를 이용하여 재실행하여 데이터베이스를 복원하는 방법
  - 백업본과 로그 필요

재해적 실패(미디어 장애 등)에 대한 회복 | 비재해적 실패에 대한 회복

- 데이터베이스 내용 자체는 손상되지 않았으나 변경 중 또는 변경된 내용에 대한 신뢰성을 잃어버린 경우 로그를 이용하여 모든 변경을 취소시켜 원래의 데이터베이스 상태로 복원하는 경우
  - 트랜잭션 장애나 시스템 장애 시 사용
  - 로그 필요

회복의 기본 원리

회복의 원리

04 회복관리자

DBMS의 서브 시스템

DBMS 코드의 10% 이상 차지

신뢰성 있는 회복을 책임

기능

- ☑ 장애 탐지
- ☑ 데이터베이스 복원

05 회복 작업

01 손상된 부분만을 포함하는 최소 단위

02 최단시간 내 작업

03 트랜잭션 기반의 회복

04 회복 자료의 보장

05 시스템 레벨의 자동 조치

로그

읽기/쓰기 연산

01 저장 장치의 유형

소멸 저장 장치 (Volatile Storage)
- 시스템의 붕괴와 함께 저장된 데이터를 상실
  예) 메인 메모리

비소멸 저장 장치 (Nonvolatile Storage)
- 시스템 붕괴 시에도 보통 저장된 데이터는 손실되지 않음
- 저장 장치 자체의 고장으로 손실 가능
  예) 디스크나 테이프

안정 저장 장치 (Stable Storage)
- 데이터의 손실이 발생하지 않게 여러 개의 비소멸 저장 장치로 구성된 저장 장치

로그

읽기/쓰기 연산

02 디스크와 메인 메모리(데이터베이스 시스템 또는 운영체재) 사이의 이동

디스크의 속도	:	기계적 속도	:	ms 단위
메인 메모리의 속도	:	전자적 속도	:	ns 단위

- 디스크에서 데이터 아이템 하나씩 가지고 오면 매우 느림
- 디스크에서 메인 메모리로 데이터를 가지고 올 때 **블록(또는 페이지) 단위**로 가지고 옴

03 Input(Bi)과 Outpt(Bi)

Input(Bi)

- 데이터 Bi가 포함되어 있는 디스크 블록을 메인 메모리에 가지고 옴
- 요청에 의함(On Demand)

Output(Bi)

- 데이터 Bi를 포함한 버퍼 블록을 디스크블록으로 이동시켜서 기록함
- 버퍼 관리자에 의하여 처리됨 (By Buffer Manager)

로그

읽기/쓰기 연산

04 프로그램과 데이터베이스(또는 OS) 사이의 데이터 이동

READ(X):X를 읽음

- 데이터 아이템 X가 버퍼 블록에 없으면 X가 포함된 블록 B_x에 대해 Input(B_x)을 실행
- 버퍼 블록에 있는 데이터 아이템 X의 값을 읽음

WRITE(X):X를 기록

- X값을 버퍼 블록에 있는 데이터 아이템 X에 기록
- 만일 데이터 아이템 X가 버퍼 블록에 없으면 X가 포함된 블록 B_x에 대해 Input(B_x)을 실행
- X값을 버퍼 블록에 있는 데이터 아이템 X에 기록

X를 포함한 버퍼블럭(B_x)의 강제 출력(Force-output)

- 어쩔 수 없이 수행하는 Output(B_x)
- 메인 메모리의 버퍼 블록에서 버퍼 관리자가 메모리 공간을 필요로 할 때 이용

READ(X)

- 어떤 트랜잭션이 제일 처음 데이터 아이템 X에 접근
- 이때 내부적으로 필요에 따라서 Input(B_x)을 실행

로그

읽기/쓰기 연산

04 프로그램과 데이터베이스(또는 OS) 사이의 데이터 이동

WRITE(X)

- 트랜잭션이 X에 대한 연산을 모든 마친 뒤에 변경된 X의 값을 데이터베이스 자체에 반영
- Write(X)를 위한 Output(B_x)는 보통 즉각 실행되지 않음
- Write(X) 연산은 실행되었지만 실제로 Output(B_x)가 실행되기 전에 시스템이 붕괴함
  - 변경된 X값의 상실

05 즉시 갱신 VS 지연 갱신

즉시 갱신		지연 갱신
Write(X) 실행 시 Output(X) 시행	VS	Write(X) 실행된 후, Force-Output에 의하여 Output(X) 시행

로그

읽기/쓰기 연산

06 WRITE(X)에 의한 페이지 기록

변경된 페이지는 언제 디스크에 기록될 것인가?

[Steal/No-Steal 방식]

Steal 방식	• 트랜잭션이 완료하기 전에 그 트랜잭션에 갱신된 캐쉬 페이지를 디스크에 기록하는 방식
No-Steal 방식	• 트랜잭션이 완료될 때까지 그 트랜잭션에 의해 갱신된 캐쉬 페이지를 디스크에 기록할 수 없는 방식

[Force/No-Force 방식]

Force 방식	• 트랜잭션이 완료할 때에 그 트랜잭션에 의해 갱신된 캐쉬 페이지를 즉시 디스크에 기록하는 방식
No-Force 방식	• 트랜잭션이 완료할 때에 그 트랜잭션에 의해 갱신된 캐쉬 페이지를 즉시 디스크에 기록할 수 없는 방식

전형적인 데이터베이스 시스템들은 Steal/No-force 방식을 사용

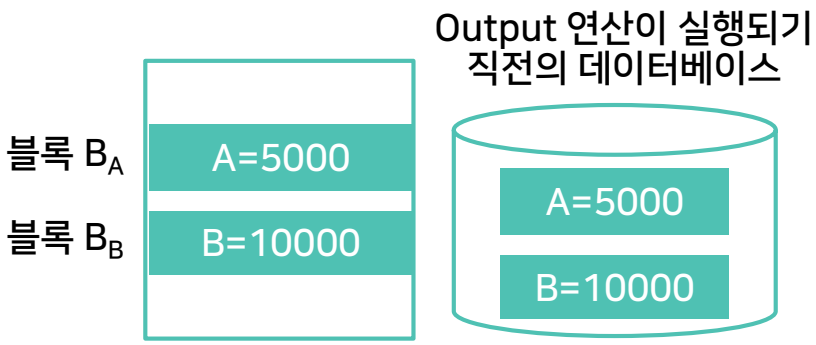
로그

읽기/쓰기 연산

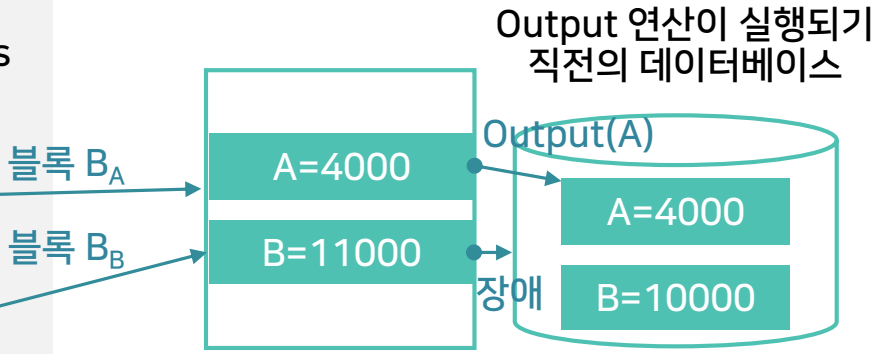
07 트랜잭션 T

예 | 계좌 A에서 1000원을 계좌 B로 이체하는 경우(A=5000, B=10000)

```
T: Begin_Trans
    Read(A)
A: = A - 1000
    Write(A)
    Read(B)
B: = B + 1000
    Write(B)
    End_Trans
```




```
T: Begin_Trans
    Read(A)
A: = A - 1000
    Write(A)
    Read(B)
B: = B + 1000
    Write(B)
    End_Trans
```



장애의 결과로 1000을 상실 → 비일관성

로그

읽기/쓰기 연산

08 제자리 갱신 VS 간접 갱신

제자리 갱신		간접 갱신
Output(X) 시행 시 Bx가 위치했던 디스크 위치에 그대로 덮어쓰기	VS	Output(X) 시행 시 Bx가 위치했던 디스크 위치와는 다른 곳에 쓰기

로그

트랜잭션

01 트랜잭션 회복

트랜잭션: 회복을 위한 논리적인 단위

02 트랜잭션의 원자성을 위한 연산

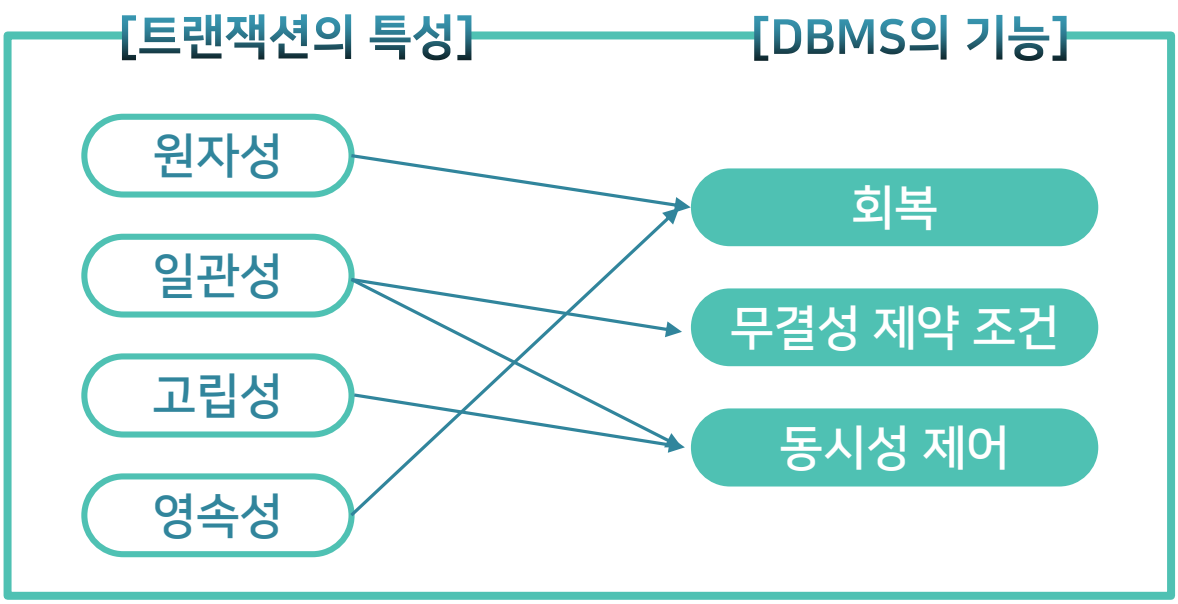
Commit
- 트랜잭션의 성공적인 실행
- 영구적 갱신
- 갱신된 데이터의 영속성을 보장

Roll Back
- 트랜잭션의 철회

로그

트랜잭션

03 트랜잭션의 특성과 DBMS의 기능

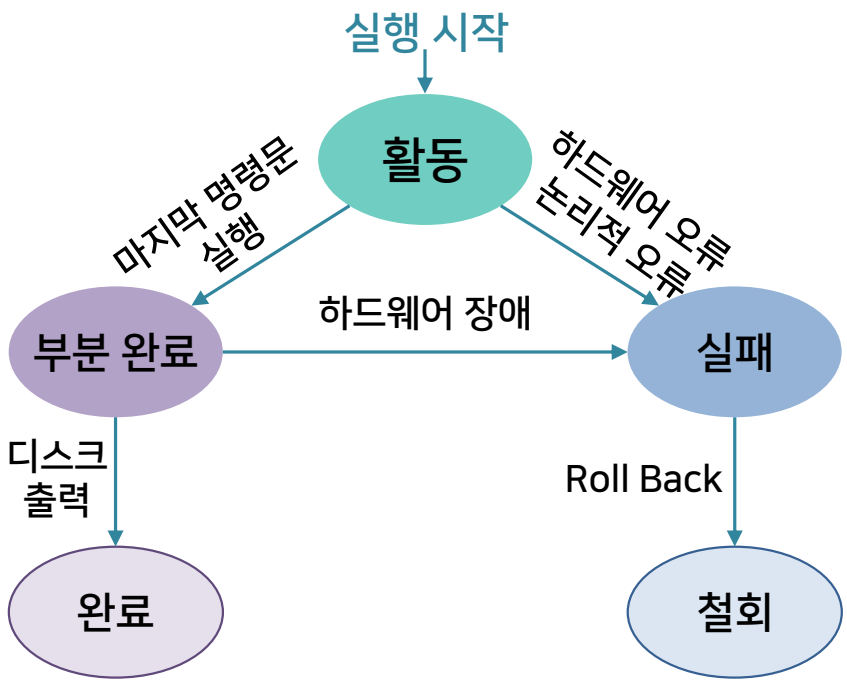


로그

트랜잭션

04 트랜잭션의 상태

부분 완료	• Commit 명령문을 받은 상태
완료	• 모든 트랜잭션 결과를 DB에 반영한 상태
실패	• 트랜잭션의 실패
철회	• Roll Back 명령문에 의하여 트랜잭션의 모든 결과를 원상태로 돌려 놓은 상태



로그

트랜잭션

05 실패 상태에 있는 트랜잭션

데이터베이스 상태를 트랜잭션 실행이 시작되기 직전의 상태로 환원시키기 위해 Roll Back 연산 실행 그 뒤에서 철회 상태의 트랜잭션으로 되어 종료됨

- 실패 상태에 있는 트랜잭션이 있을 경우 취할 수 있는 조치

01 트랜잭션의 재시작

- 하드웨어나 시스템 소프트웨어 오류로 인해 철회된 트랜잭션은 다시 새로운 트랜잭션으로 취급되어 재시작 됨

02 트랜잭션의 폐기

- 트랜잭션의 내부적 오류로 재작성을 하던지, 원하는 데이터가 데이터베이스에 없는 경우에 취하는 조치

로그

트랜잭션

06 로그의 기본 구조

로그 레코드

```
<Ti, Start>              : 트랜잭션 Ti의 시작
<Ti, X, v_old, v_new> : Ti가 데이터 아이템 X의 값 v_old를 v_new로 변경
  ...
<Ti. commit>             : 트랜잭션 Ti의 실행 완료
```

UNDO 연산	• 로그 레코드에 기록된 연산 취소 • OLD 값으로 복원
REDO 연산	• 로그 레코드에 기록된 연산들을 다시 수행 • 새로운 값으로 복원

로그

WAL 규약

01 로그 우선 기록 규약(Write Ahead Log)

- 로그 기반의 회복을 위한 기본 규약
- 데이터 아이템을 디스크에 쓰기 전에 대응되는 로그 정보를 먼저 써야 한다는 규약
  - 디스크 상의 항목의 이전 값을 이후 값으로 덮어쓰기 전에 데이터 항목의 이전 값을 포함하는 로그 엔트리가 디스크로 강제 출력되는 것을 보장하는 방식
  - 트랜잭션을 위한 모든 REDO 유형과 UNDO 유형 로그 기록들이 디스크로 강제 출력된 후에야 트랜잭션이 완료될 수 있음

트랜잭션Ti의 출력	<Ti ,commit> 로그 레코드를 안정 저장 장치에 출력시켜야만 완료상태로 들어갈 수 있음
<Ti, commit> 로그 레코드의 출력	먼저 Ti에 관련된 모든 로그 레코드를 안정 저장 장치에 출력시켜야 됨
데이터베이스 버퍼 블록의 출력	먼저 버퍼 블록의 데이터와 관련된 모든 로그 레코드가 안정 저장 장치에 출력되어야 함

1 회복의 기본 원리

- ✓ 장애: 시스템이 정해진 명세대로 작동하지 않는 상태
- ✓ 회복: 장애가 일어났을 때 장애 발생 이전 상태로 복원시키는 것
- ✓ 회복의 기본 원리: 중복

2 로그

- ✓ 로그: 트랜잭션의 연산 내용을 기록해 놓은 곳 <트랜잭션ID, 변경 전 값, 변경 후 값>
- ✓ WAL: 트랜잭션의 연산 결과를 DB에 기록하기 전에 로그에 먼저 기록해야 한다는 규약