트랜잭션, 동시성 제어, 회복

목차

회복(Recovery)

학습내용

- 트랜잭션과 회복
- 로그 파일
- 로그 파일을 이용한 회복
- 체크포인트를 이용한 회복

회복(recovery)

- DB 장애 시 DB를 일관성 있는 상태로 되돌리는 DBMS 기능
- the process of restoring a database to the correct state in the event of a failure.
- It ensures that the database is reliable and remains in consistent state in case of a failure.

회복(recovery)-DB 장애 유형

- 시스템 사고(System Crash)
  - 하드웨어 혹은 소프트웨어 오류로 인한 주기억장치 손실
  - 주기억장치 상주, 처리 중인 프로그램과 데이터의 손실
- 미디어 장애(Media Failure)
  - 보조기억장치의 일부 데이터가 손실
    - 헤드 충돌이나 읽기 장애에 의하여
  - 보조기억장치에 저장 중인 데이터 손실

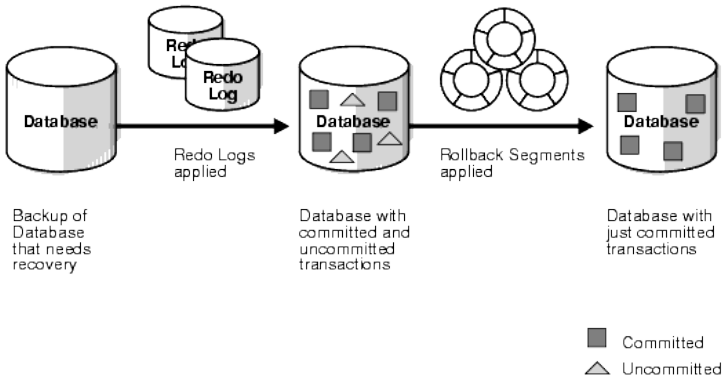
회복(recovery)–DB 장애 유형

- 응용 소프트웨어 오류(App. Error)
  - DB에 접근하는 소프트웨어의 논리적인 오류로 트랜잭션 수행이 실패
- 자연재해(Natural Disaster)
  - 화재, 홍수, 지진, 정전 등에 의해 컴퓨터 시스템 손상
- 부주의 혹은 태업(Sabotage)
  - 운영자나 사용자의 부주의로 데이터 손실 또는 의도적인 손상을 입음

Basic Recovery Steps: Rolling Forward and Rolling Back

- Rollback:
  - Undo any partially completed transactions (ones in progress when the crash occurred) by applying the before images to the database.
- Rollforward:
  - Redo the transactions by applying the after images to the database.
  - This is done for transactions that were committed before the crash.

Basic Recovery Steps: Rolling Forward and Rolling Back



트랜잭션과 회복

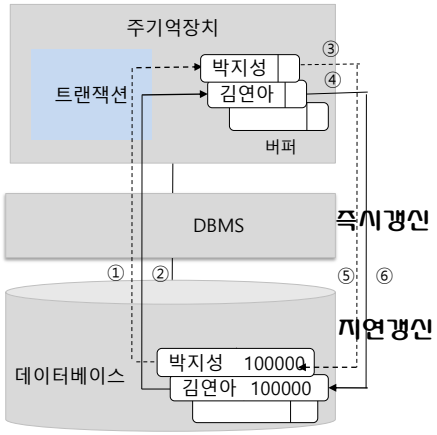
- 트랜잭션은 회복의 단위
- 회복관리자(Recovery Manager)
  - 장애로부터 데이터베이스 보호
  - 트랜잭션의 원자성과 지속성을 보장
- 로그(transaction log::database log::binary log)
  - a history of actions executed by DBMS
  - Physically, a file listing changes to the database, stored in a stable storage format.

트랜잭션과 회복

```
START TRANSACTION
① /* 박지성 계좌를 읽어온다 */
② /* 김연아 계좌를 읽어온다 */
  /* 잔고 확인 */

③ /* 예금인출 박지성 */
UPDATE Customer
SET balance=balance-10000
WHERE name= '박지성' ;

④ /* 예금입금 김연아 */
UPDATE Customer
SET balance=balance+10000
WHERE name= '김연아' ;

COMMIT /* 부분완료 */
⑤ /* 박지성 계좌를 기록한다 */
⑥ /* 김연아 계좌를 기록한다 */
```

(a) 계좌이체 트랜잭션

(DBMS가 즉시 갱신하는 경우 COMMIT전에 ⑤⑥이 DB에 기록될 수도 있음)

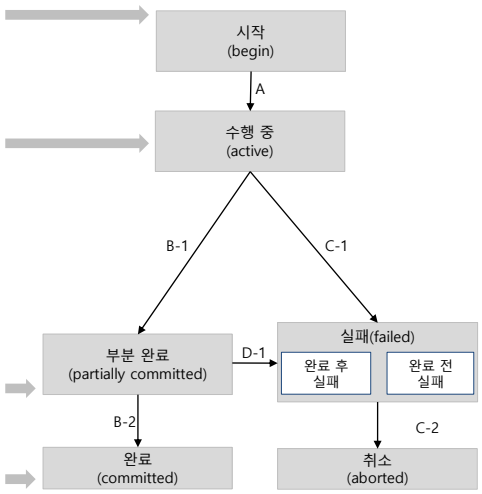


(b) 트랜잭션 수행과정

트랜잭션과 회복:트랜잭션 수행과 상태도

```
BEGIN TRANSACTION
① /* 박지성 계좌를 읽어온다 */
② /* 김연아 계좌를 읽어온다 */
  /* 잔고 확인 */

③ /* 예금인출 박지성 */
UPDATE Customer
SET balance=balance-100
WHERE name='박지성';

④ /* 예금입금 김연아 */
UPDATE Customer
SET balance=balance+100
WHERE name='김연아';

COMMIT /* 부분완료 */
⑤ /* 박지성 계좌를 기록한다 */
⑥ /* 김연아 계좌를 기록한다 */
```

(a) 계좌이체 트랜잭션



(b) 트랜잭션 상태도

로그 파일(log file)

- 트랜잭션의 데이터베이스 기록을 추적
- 목적
  - DBMS는 트랜잭션이 수행 중이거나 수행이 종료된 후 발생하는 데이터베이스 손실을 방지
- 트랜잭션이 반영한 모든 데이터의 변경사항을 데이터베이스에 기록하기 전에 미리 기록해두는 별도의 데이터베이스
- 안전한 하드디스크에 저장됨
- 전원과 관계없이 기록이 남아있음

로그의 구조

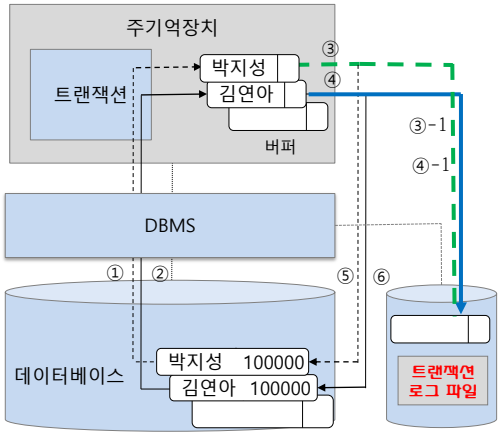
- <트랜잭션번호, 로그의 타입, 데이터 항목 이름, 수정 전 값, 수정 후 값>
- 로그의 타입
  - 트랜잭션의 연산 타입
  - START, INSERT, UPDATE, DELETE, ABORT, COMMIT 등
- 수정 전 값
  - 데이터의 변경 전 값
- 수정 후 값
  - 연산의 결과로 변경된 값

<T1, START>
<T1, UPDATE, Customer(박지성).balance, 100000, 90000>
<T1, UPDATE, Customer(김연아).balance, 100000, 110000>
<T1, COMMIT>

트랜잭션 수행과 로그 파일

```
START TRANSACTION
① /* 박지성 계좌를 읽어온다 */
② /* 김연아 계좌를 읽어온다 */
   /* 잔고 확인 */

③ /* 예금인출 박지성 */
UPDATE Customer
SET balance=balance-10000
WHERE name='박지성';

④ /* 예금입금 김연아 */
UPDATE Customer
SET balance=balance+10000
WHERE name='김연아';

COMMIT /* 부분완료 */
⑤ /* 박지성 계좌를 기록한다 */
⑥ /* 김연아 계좌를 기록한다 */
```

(a) 계좌이체 트랜잭션



(b) 트랜잭션 수행과정

로그 파일을 이용한 회복

- 두 개의 트랜잭션이 실행된다고 가정
- 트랜잭션 연산
  - SELECT는 read_item( )
  - UPDATE는 write_item()
- 트랜잭션은 각각 데이터 A, B, C, D를 접근
  - 읽거나 쓰는 작업을 진행
- 데이터(A, B, C, D)의 초기값
  - (100, 200, 300, 400)

로그 파일을 이용한 회복

트랜잭션 T1	트랜잭션 T2
read_item(A);	read_item(A);
A=A+10;	A=A+10;
read_item(B);	write_item(A);
B=B+10;	read_item(D);
write_item(B);	D=D+10;
read_item(C);	read_item(B)
C=C+10;	B=B+10;
write_item(C);	write_item(B);
write_item(A);	write_item(D);

(A, B, C, D):
(100, 200, 300, 400)

로그 파일을 이용한 회복

✓ 트랜잭션이 T1 → T2 순으로 실행된다면 다음과 같은 로그 파일이 생성된다.

로그 번호	로그 레코드
1	[T1, START]
2	[T1, UPDATE, B, 200, 210]
3	[T1, UPDATE, C, 300, 310]
4	[T1, UPDATE, A, 100, 110]
5	[T1, COMMIT]
6	[T2, START]
7	[T2, UPDATE, A, 110, 120]
8	[T2, UPDATE. B, 210, 220]
9	[T2, UPDATE, D, 400, 410]
10	[T2, COMMIT]

(A, B, C, D):
(100, 200, 300, 400)

트랜잭션 T1	트랜잭션 T2
read_item(A);	read_item(A);
A=A+10;	A=A+10;
read_item(B);	write_item(A);
B=B+10;	read_item(D);
write_item(B);	D=D+10;
read_item(C);	read_item(B)
C=C+10;	B=B+10;
write_item(C);	write_item(B);
write_item(A);	write_item(D);

로그 파일을 이용한 회복

- 시스템 운영 중 장애가 발생하여 시스템이 다시 가동되었을 때
  - DBMS는 로그 파일을 먼저 검토
- DBMS는 트랜잭션이 종료되었는지 혹은 중단되었는지 여부를 판단
  - 종료된 트랜잭션은 종료를 확정하기 위하여 **재실행(REDO)**을 진행
  - 중단된 트랜잭션은 없던 일로 되돌리기 위해 **취소(UNDO)**를 진행

트랜잭션의 재실행(REDO)

- 장애가 발생한 후 시스템을 다시 가동을 했을 때,
- 로그파일에 트랜잭션 시작(START)과 종료(COMMIT)가 있는 경우
  - COMMIT 연산이 로그에 있다는 것은 트랜잭션이 모두 완료되었다는 의미
- 변경 내용이 버퍼에서 DB에 기록되지 않았을 가능성이 있음
- 따라서 로그를 보면서 트랜잭션이 변경한 내용을 DB에 다시 기록하는 과정이 필요.
  - 이 과정을 **REDO**라고 함

트랜잭션의 취소(UNDO)

- 장애가 발생한 후 시스템을 다시 가동했을 때,
- 로그 파일에 트랜잭션의 시작(START)만 있고 종료(COMMIT)가 없는 경우
  - COMMIT 연산이 로그에 없는 것: 트랜잭션이 완료되지 못했다는 의미
- 트랜잭션이 한 일을 모두 취소해야 함
- 이 경우 완료하지 못했지만 버퍼의 변경 내용이 DB에 기록되어 있을 가능성이 있음
- 따라서 로그를 보면서 트랜잭션이 변경한 내용을 DB에서 원상 복구시켜야 함.
  - 이 과정을 UNDO라고 함

로그 파일을 이용한 회복

- 즉시 갱신 방법
  - Immediate or in-place update
  - '갱신 데이터→로그', '버퍼→데이터베이스' 작업이 부분완료 전에 동시에 진행될 수 있음
  - 부분완료가 되면 갱신 데이터는 로그에 기록이 끝난 상태
- 지연 갱신 방법(deferred update)
  - '갱신 데이터→로그'가 끝난 후 부분완료를 하고
  - '버퍼→데이터베이스' 작업이 진행되는 방법

즉시/지연 갱신 로그파일 예

✓ 트랜잭션이 T1 → T2 순으로 실행된다면 다음과 같은 로그 파일이 생성된다.

로그 번호	로그 레코드
1	[T1, START]
2	[T1, UPDATE, B, 200, 210]
3	[T1, UPDATE, C, 300, 310]
4	[T1, UPDATE, A, 100, 110]
5	[T1, COMMIT]
6	[T2, START]
7	[T2, UPDATE, A, 110, 120]
8	[T2, UPDATE. B, 210, 220]
9	[T2, UPDATE, D, 400, 410]
10	[T2, COMMIT]

트랜잭션 T1	트랜잭션 T2
read_item(A);	read_item(A);
A=A+10;	A=A+10;
read_item(B);	write_item(A);
B=B+10;	read_item(D);
write_item(B);	D=D+10;
read_item(C);	read_item(B)
C=C+10;	B=B+10;
write_item(C);	write_item(B);
write_item(A);	write_item(D);

즉시 갱신 방법

로그 번호	작업	결과
i =0	아무 작업도 필요 없음	T1과 T2가 수행을 시작하지 않았음
1 < = i < = 4	UNDO(T1) : T1을 취소 → T1이 i까지 생성한 로그 레코드를 이용하여 데이터베이스 항목을 되돌림	T1을 수행하지 않은 것과 같음
5 < = i < = 9	REDO(T1) : T1을 재수행 → 1부터 4까지 T1이 생성한 로그 레코드를 이용하여 데이터베이스 항목 값을 기록함 UNDO(T2) : T2를 취소 → T2가 5부터 i까지 생성한 로그 레코드를 이용하여 데이터베이스 항목을 되돌림	T1은 수행이 완료됨 T2는 수행하지 않은 것과 같음
10	REDO(T1) : T1을 재수행 REDO(T2) : T2를 재수행	T1, T2는 수행이 완료됨

지연 갱신 방법

로그 번호	작업	결과
i =0	아무 작업도 필요 없음	T1과 T2가 수행을 시작하지 않았음
1 < = i < = 4	T1 : 아무 작업도 필요 없음	T1을 수행하지 않은 것과 같음
5 < = i < = 9	REDO(T1) : T1을 재수행 → 1부터 4까지 T1이 생성한 로그 레코드를 이용하여 데이터베이스 항목 값을 기록함 T2 : 아무 작업도 필요 없음	T1은 수행이 완료됨 T2는 수행하지 않은 것과 같음
10	REDO(T1) : T1을 재수행 REDO(T2) : T2를 재수행	T1, T2는 수행이 완료됨

체크포인트를 이용한 회복

- 로그를 이용한 회복
  - 시스템 장애 시 어느 시점까지 되돌아가야 하는지 알 수 없음
- 트랜잭션이 많은 응용의 경우
  - 하루 이상 되돌아가서 복구하는 것은 사실상 불가능
- 체크포인트(checkpoint, 혹은 검사점)
  - 회복 시 많은 양의 로그를 검색하고 갱신하는 시간을 단축 위함
  - 몇 십 분 단위로 DB와 트랜잭션 로그 파일을 동기화한 후
  - 동기화한 시점을 로그 파일에 기록해두는 방법 혹은 그 시점

체크포인트 시점의 작업 진행

- 주기억장치의 로그 레코드를 모두 하드디스크의 로그 파일에 저장
- 버퍼에 있는 변경된 내용을 하드디스크의 데이터베이스에 저장 ⇒ 즉시갱신경우
- 체크포인트를 로그 파일에 표시

체크포인트를 이용한 회복

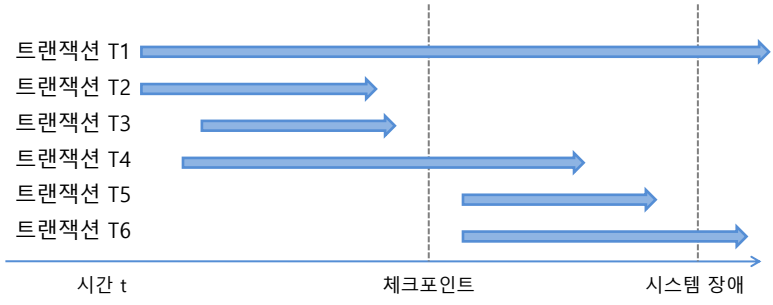
- 체크포인트가 있으면 로그를 이용한 회복 기법은 좀더 간단
- 체크포인트 이전에 [COMMIT] 기록이 있는 경우
  - 아무 작업이 필요 없음.
  - 로그에 체크포인트가 나타나는 시점은 이미 변경 내용이 데이터베이스에 모두 기록된 후이기 때문.
- 체크포인트 이후에 [COMMIT] 기록이 있는 경우
  - REDO(T)를 진행.
  - 체크포인트 이후에 변경 내용이 데이터베이스에 반영되지 않았으므로 REDO를 진행.

체크포인트를 이용한 회복

- 체크포인트 이후에 [COMMIT] 기록이 없는 경우
  - 즉시갱신방법을 사용했다면 UNDO(T)를 진행.
    - 버퍼 내용이 반영됐을 수도 있기 때문에 원상복구 시켜야 함.
  - 지연갱신방법을 사용했다면 아무것도 할 필요가 없음.
    - 지연 갱신 방법은 [COMMIT] 이전에는 버퍼의 내용을 데이터베이스에 반영하지 않기 때문.

체크포인트를 이용한 회복

- 즉시 갱신 방법 사용
  - T2, T3는 아무 작업이 필요 없고, T4, T5는 REDO, T1, T6는 UNDO가 필요함.
- 지연 갱신 방법 사용
  - T2, T3는 아무 작업이 필요 없고, T4, T5는 REDO가 필요함. T1, T6는 아무 작업이 필요 없음.



체크포인트가 포함된 로그 기록

- 트랜잭션 T1, T2, T3가 동시에 실행된 후 다음과 같이 로그 기록을 남김
- 즉시 갱신 기법을 사용하여 회복을 한다면 REDO(T2), UNDO(T3)가 진행됨
- T1에 대해서는 아무 작업이 필요 없음

로그 번호	로그 레코드
1	[T1, START]
2	[T1, UPDATE, B, 200, 120]
3	[T1, UPDATE, C, 300, 310]
4	[T2, START]
5	[T2, UPDATE, A, 110, 120]
6	[T1, UPDATE, A, 120, 110]
7	[T1, COMMIT]
8	[T2, UPDATE. B, 120, 220]
9	[CHECKPOINT]
10	[T3, START]
11	[T3, UPDATE, A, 110, 120]
12	[T2, UPDATE, D, 400, 410]
13	[T2, COMMIT]
14	[T3, UPDATE. B, 220, 230]
15	~~ 시스템 장애 ~~

체크포인트가 포함된 로그 기록

- 트랜잭션 T1, T2, T3가 동시에 실행된 후 다음과 같이 로그 기록을 남김
- 지연 갱신 기법을 사용하여 회복을 한다면 REDO(T2)가 진행됨
- T1, T3에 대해서는 아무 작업이 필요 없음

로그 번호	로그 레코드
1	[T1, START]
2	[T1, UPDATE, B, 200, 120]
3	[T1, UPDATE, C, 300, 310]
4	[T2, START]
5	[T2, UPDATE, A, 110, 120]
6	[T1, UPDATE, A, 120, 110]
7	[T1, COMMIT]
8	[T2, UPDATE. B, 120, 220]
9	[CHECKPOINT]
10	[T3, START]
11	[T3, UPDATE, A, 110, 120]
12	[T2, UPDATE, D, 400, 410]
13	[T2, COMMIT]
14	[T3, UPDATE. B, 220, 230]
15	~~ 시스템 장애 ~~


ENJOY YOUR DB!!!


THANK YOU
또 보고 싶을 거야
Coming Soon...