1 직렬 스케줄

01 직렬 스케쥴

1 스케쥴 개요

- 트랜잭션들이 인터리빙 방식으로 동시에 수행될 때 여러 트랜잭션들이 가진 연산들의 실행 순서를 스케줄(또는 히스토리)이라 함

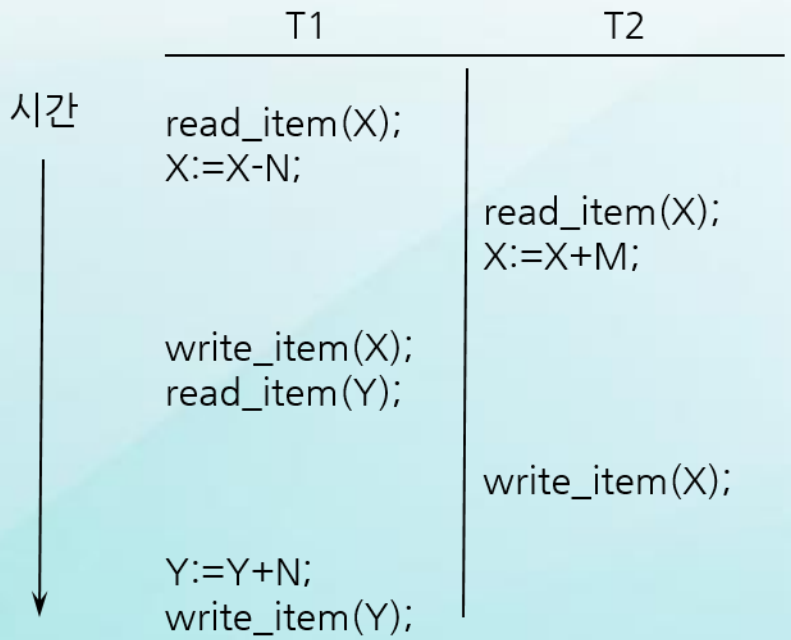
- n개의 트랜잭션 T1, T2, ..., Tn 의 스케줄 S 는 이들 트랜잭션들의 연산들을 순서화한 것임

01 직렬 스케쥴

1 스케쥴 개요

여기서 스케줄 S 에 참여하는 각 트랜잭션 Ti 에 대해서 Ti 의 연산들이 Ti 내에서와 동일한 순서로 스케줄 S 에 나타나야 함

- (예) S
  : r1(X); r2(X); w1(X);
  r1(Y); w2(X); w1(Y);



※ 출처 : 데이터베이스 시스템 6판, Elmasri, Navathe 저, 황규영 외 역, 홍릉과학출판사, 2016년

01 직렬 스케쥴

1 스케쥴 개요

- 연산 충돌(Conflict)
  - 다수의 트랜잭션들이 같은 스케쥴 안에서 병행처리 될 때 갱신 손실, 오손 읽기, 반복할 수 없는 읽기, 부정확한 요약 등의 문제를 초래할 수 있는 트랜잭션간 연산의 충돌을 의미

01 직렬 스케쥴

1 스케쥴 개요

- 연산 충돌(Conflict)
  - 하나의 스케줄에 포함된 두 개의 연산이 아래의 세 가지 조건들을 모두 만족하면 충돌(Conflict)이라고 함
    (1) 그 연산들은 서로 다른 트랜잭션에 속함
    (2) 그 연산들은 동일한 항목 X 에 접근함
    (3) 그 연산들 중 최소한 하나는 write_item(X)

01 직렬 스케쥴

1 스케쥴 개요

회복가능한 스케줄 (Recoverable Schedule)

- 트랜잭션 T 가 완료되었다면 절대로 복귀(Rollback) 될 필요가 없다는 기준을 두고, 이 기준을 만족하는 스케줄을 회복가능한 스케줄이라 함

회복가능(Recoverable)

- 스케줄 S 내의 어떤 트랜잭션 T 에 대해서도 T 가 읽은 항목에 쓰기연산을 수행한 모든 트랜잭션 T' 이 완료되기 전까지 T 가 완료되지 않는다면 S 가 회복가능하다고 함

01 직렬 스케쥴

1 스케쥴 개요

- 연쇄복귀(Cascading Rollback) /연쇄철회(Cascading Abort)
  - 완료되지 않은 트랜잭션이 실패한 트랜잭션으로 부터 항목을 읽어옴으로써 발생하는 것으로 하나의 트랜잭션 철회가 다른 트랜잭션의 철회로 이어지는 현상이 발생함
  - (예) S : r1(X); w1(X); r2(X);
    r1(Y); w2(X); w1(Y); a1; a2

01 직렬 스케쥴

1 스케쥴 개요

- 연쇄복귀방지
  - 완료된 트랜잭션이 쓴 항목만을 읽음

- 엄격한 스케줄(Strict Schedule)
  - 마지막으로 X 에 쓰기연산을 한 트랜잭션이 완료되거나 철회되기 전까지 다른 트랜잭션이 X 를 읽지도 쓰지도 못함
  - 회복과정 : 철회된 쓰기연산이 적용되기 전의 X 가 가지고 있던 이전값(BFIM; Before Image)으로 복귀함

01 직렬 스케쥴

1 스케쥴 개요

- 회복 가능성을 근거로 지금까지 스케줄들을 (1) 회복가능성, (2) 연쇄 복귀의 방지, (3) 엄격함이라는 관점에 따라 특성화하였음

- 스케줄들이 가진 특성들은 번호가 높아짐에 따라 더 엄중한 조건임을 알 수 있음, 그러므로 조건 (2)는 조건 (1)을 암시하고 조건 (3)은 조건 (2)와 (1) 모두를 암시함

- 따라서, 모든 엄격한 스케줄들은 연속적인 철회가 필요 없으며, 모든 연속적인 철회가 필요 없는 스케줄들은 회복가능함

01 직렬 스케쥴

2 직렬 스케쥴 개요

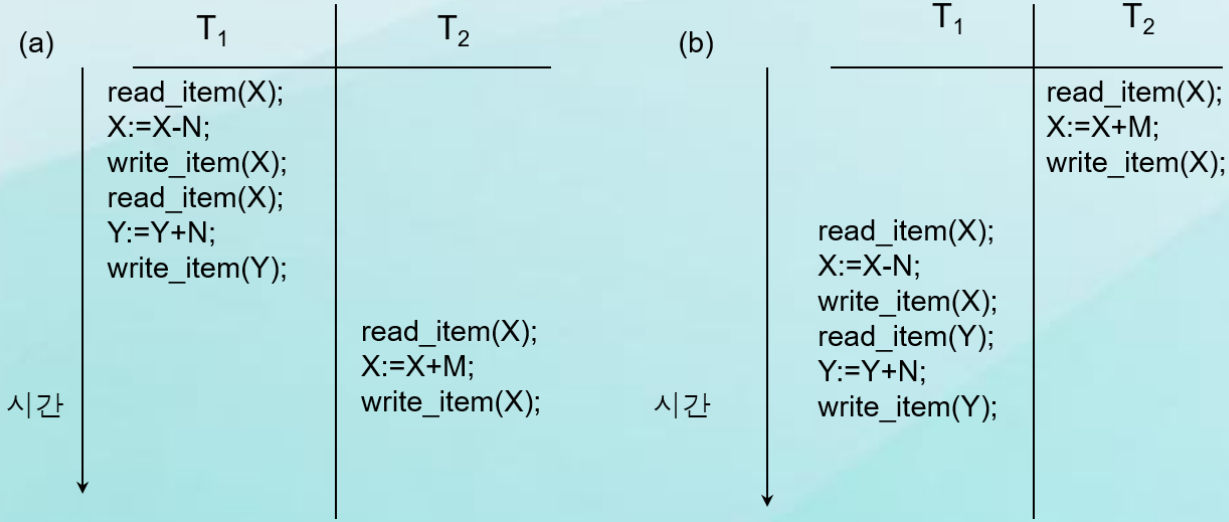
- 직렬 스케줄
  : 스케줄에 참가하는 각 트랜잭션 T 에 대해서 T 에 속한 모든 연산들이 다른 트랜잭션의 연산들과 인터리빙되지 않고 연속적으로 실행될 때 직렬 스케줄이라고 함

- 스케줄의 직렬가능
  : n 개 트랜잭션들로 구성된 스케줄 S가 동일한 n 개의 트랜잭션들로 구성된 어떤 직렬 스케줄과 동치일 경우를 의미함

01 직렬 스케쥴

2 직렬 스케쥴 개요

직렬 스케쥴의 예
: 다른 트랜잭션과 인터리빙 없이
연속적으로 수행되므로 직렬 스케줄임

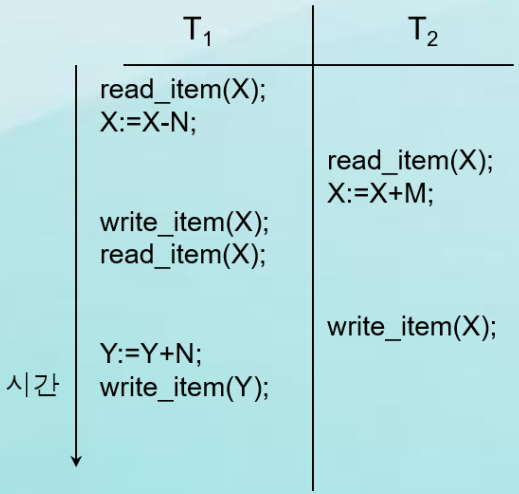


01 직렬 스케쥴

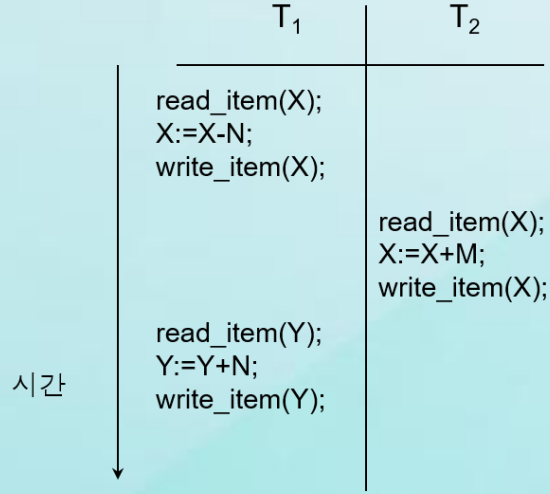
2 직렬 스케쥴 개요

비직렬 스케쥴의 예
: 두 트랜잭션들의 연산들이 인터리빙되므로 비직렬 스케줄임

(c)



(d)



2 직렬 가능 스케줄

02 직렬 가능 스케쥴

1 직렬가능 스케쥴 개요

- 충돌 연산이 있음에도 불구하고 직렬 스케쥴과 동치인 스케쥴을 충돌 직렬가능 스케쥴(Conflict Serializable Schedule)이라 함

- 충돌 연산

  (1) Read - Write (RW)
  (2) Write - Read (WR)
  (3) Write - Write (WW)

- 직렬가능 스케쥴이 되기 위해서는 충돌의 순서가 유지되어야 함

02 직렬 가능 스케쥴

2 직렬가능 테스트 알고리즘

1. 스케쥴 S 에 참가하는 각 트랜잭션 Ti 에 대해 선행 그래프(Precedence Graph)에 Ti 라는 레이블을 가진 노드를 생성

2. 스케줄 S 에서 Ti가 read_item(X) 를 실행한 후에 Tj 가 write_item(X)를 실행하는 경우마다 선행 그래프에 간선 (Ti → Tj) 을 생성함

3. 스케줄 S 에서 Ti 가 write_item(X) 를 실행한 후에 Tj 가 read_item(X) 를 실행하는 경우마다 선행 그래프에 간선 (Ti → Tj) 을 생성함

02 직렬 가능 스케쥴

2 직렬가능 테스트 알고리즘

4 스케줄 S 에서 Ti 가 write_item(X) 를 실행한 후에 Tj 가 write_item(X)를 실행하는 경우마다 선행 그래프에 간선 (Ti → Tj) 을 생성함

5 스케줄 S 가 직렬 가능하다는 것은 선행 그래프에 사이클이 없다는 것의 필요충분조건임

02 직렬 가능 스케쥴

2 직렬가능 테스트 알고리즘

충돌 직렬성 테스트 예제

- 스케쥴에 포함되는 트랜잭션들

transaction T1

read_item(X);
write_item(X);
read_item(Y);
write_item(Y);

transaction T2

read_item(Z);
read_item(Y);
write_item(Y);
read_item(X);
write_item(X);

transaction T3

read_item(Y);
read_item(Z);
write_item(Y);
write_item(Z);

02 직렬 가능 스케쥴

2 직렬가능 테스트 알고리즘

- 충돌 직렬성 테스트 예제
  - 스케쥴 S1

T1	T2	T3
	read_item(Z);	
	read_item(Y);	
	write_item(Y);	
		read_item(Y);
		read_item(Z);
read_item(X);		
write_item(X);		
		write_item(Y);
		write_item(Z);
	read_item(X);	
read_item(Y);		
write_item(Y);		
	write_item(X);	

02 직렬 가능 스케쥴

2 직렬가능 테스트 알고리즘

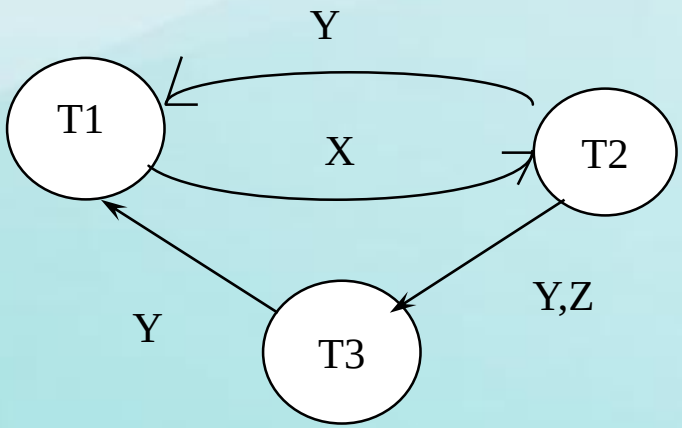
- 충돌 직렬성 테스트 예제
  - 스케쥴 S2

T1	T2	T3
		read_item(Y);
		read_item(Z);
read_item(X);		
write_item(X);		write_item(Y);
		write_item(Z);
	read_item(Z);	
read_item(Y);		
write_item(Y);		
	read_item(Y);	
	write_item(Y);	
	read_item(X);	
	write_item(X);	

02 직렬 가능 스케쥴

2 직렬가능 테스트 알고리즘

- 충돌 직렬성 테스트 예제
  - 선행 그래프(Precedence Graph)



[스케쥴 S1 의 선행 그래프]

동치인 직렬 스케줄: 없음

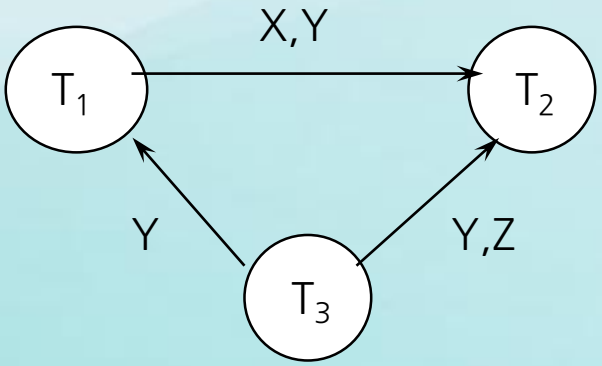
사이클 T1 → T2 → T1
사이클 T1 → T2 → T3 → T1

02 직렬 가능 스케쥴

2 직렬가능 테스트 알고리즘

충돌 직렬성 테스트 예제

- 선행 그래프(Precedence Graph)



[스케쥴 S2 의 선행 그래프]

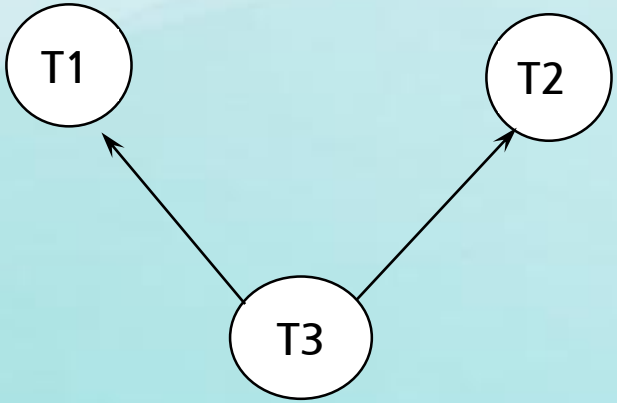
동치인 직렬 스케줄: T3 - T1 - T2

사이클이 없다

02 직렬 가능 스케쥴

2 직렬가능 테스트 알고리즘

- 충돌 직렬성 테스트 예제
  - 동치인 직렬 스케줄을 두 개 가지는 스케줄을 나타내는 선행 그래프



동치인 직렬 스케줄
T3 - T2 - T1
T3 - T1 - T2

02 직렬 가능 스케쥴

3 직렬가능 스케쥴의 용도

- 직렬가능한 스케줄에서는 어떠한 정확성도 잃지 않으면서 동시 실행의 장점을 얻을 수 있음

- 직렬 가능성 테스트의 문제점
  - 직렬가능성을 보장하기 위해 스케줄의 연산들이 어떻게 인터리빙 될 것인가를 미리 결정하는 것은 실질적으로 불가능함
  - 비실용적임 (직렬가능이 아니라고 판명되면 그 스케줄의 영향을 반드시 취소시켜야 함)

02 직렬 가능 스케쥴

3 직렬가능 스케쥴의 용도

직렬 가능성을 보장해주는 동시성 제어 프로토콜
(추후에 소개될 내용)

- 2PL(2 Phase Locking, 2 단계 로킹) 방법
- 타임스탬프(Timestamp) 순서화에 근거한 방법
- 데이타 항목의 여러 버전을 유지하는 방법
- 보증 또는 검증에 기반을 두는 낙관적 방법

3 SQL의 트랜잭션 지원

03 SQL의 트랜잭션 지원

1 SQL 에서의 트랜잭션

- SQL 트랜잭션의 정의는 앞서 살펴본 트랜잭션의 개념과 유사함

- 즉 SQL 트랜잭션은 작업의 논리적인 단위이며 원자성이 보장됨

- 하나의 SQL 문은 오류 없이 실행을 완료하든지 또는 실패하여 데이터베이스를 수정하지 않은 상태로 두든지 간에 항상 원자적 수행 단위로 간주

03 SQL의 트랜잭션 지원

1 SQL 에서의 트랜잭션

- SQL에는 명시적인 Begin_Transaction 문이 없음
- 트랜잭션은 특정한 SQL 문을 만났을 때
  묵시적으로 시작함
- 모든 트랜잭션은 명시적인 종료(end) 문을 가져야 함,
  명시적인 종료 문이란 COMMIT 혹은 ROLLBACK임

03 SQL의 트랜잭션 지원

1 SQL 에서의 트랜잭션

SQL 트랜잭션의 예

```
EXEC SQL WHENEVER SQLERROR GOTO UNDO;
EXEC SQL SET TRANSACTION
    READ WRITE
    DIAGNOSTIC SIZE 5
    ISOLATION LEVEL SERIALIZABLE;
EXEC SQL INSERT INTO EMPLOYEE (Fname, Lname, Ssn, Dno, Salary)
    VALUES ('Robert', 'Smith', '991004321', 2, 35000);
EXEC SQL UPDATE EMPLOYEE
    SET Salary = Salary * 1.1 WHERE Dno = 2;
EXEC SQL COMMIT;
GOTO THE_END;
UNDO: EXEC SQL ROLLBACK;
THE_END: ... ;
```

※ 출처 : 데이터베이스 시스템 6판, Elmasri, Navathe 저, 황규영 외 역, 홍릉과학출판사, 2016년

03 SQL의 트랜잭션 지원

2 SQL 트랜잭션의 속성

- 모든 트랜잭션들은 속성들을 가지고 있으며, 이러한 속성들은 SET TRANSACTION 문으로 명시됨

- 이러한 속성들에는 접근 모드(Access Mode), 진단 영역 크기(Diagnostic Area Size), 고립성 등급(Isolation Level)이 있음

03 SQL의 트랜잭션 지원

2 SQL 트랜잭션의 속성

트랜잭션의 속성을 지정하는 Set Transaction 문의 예

```
EXEC SQL SET TRANSACTION
        READ WRITE
        DIAGNOSTIC SIZE 5
        ISOLATION LEVEL SERIALIZABLE;
```

03 SQL의 트랜잭션 지원

2 SQL 트랜잭션의 속성

접근 모드

- READ ONLY
  - 데이타 검색만이 가능함
  - 고립성 등급을 READ UNCOMMITTED로 명시한다면 접근 모드는 READ ONLY로 됨
- READ WRITE
  - 수정, 삽입, 삭제, 생성 명령을 실행할 수 있음
  - 고립성 등급을 READ UNCOMMITTED로 명시하지 않는 한 접근 모드의 디폴트는 READ WRITE임

03 SQL의 트랜잭션 지원

2 SQL 트랜잭션의 속성

- 진단 영역 크기 : DIAGNOSTIC SIZE n
  - 정수 값 n을 지정하며 이는 진단 영역에서 동시에 유지할 수 있는 조건들의 개수를 나타냄
  - 이러한 조건들은 사용자에게 가장 최근에 실행한 SQL 문에 대하여 반환 정보(즉, 오류 혹은 예외)를 제공함

03 SQL의 트랜잭션 지원

2 SQL 트랜잭션의 속성

고립성

- DBMS의 트랜잭션의 가져야하는 특성의 하나로써
- 다수의 트랜잭션이 병행 실행되는 경우,
  어느 하나의 트랜잭션 실행 중에 다른 트랜잭션의
  연산이 끼어들 수 없도록 함
- 또는 수행중인 트랜잭션은 다른 트랜잭션의
  수행 결과를 그 트랜잭션이 완료될 때까지
  참조할 수 없도록 함을 뜻하는 용어

고립성의 정도를 강하게 하면 병행성이 떨어지고
약하게 하면 병행처리 시 발생할 수 있는 문제점들의
발생 가능성이 높아짐

03 SQL의 트랜잭션 지원

2 SQL 트랜잭션의 속성

- 고립성 등급 : ISOLATION LEVEL <isolation>
  - <isolation>의 값으로
    - READ UNCOMMITTED
    - READ COMMITTED
    - REPEATABLE READ
    - SERIALIZABLE
  - 디폴트 고립성 등급은 SERIALIZABLE이며, 몇몇 시스템에서는 READ COMMITTED를 디폴트 고립성 등급으로 사용하고 있음

03 SQL의 트랜잭션 지원

2 SQL 트랜잭션의 속성

- SERIALIZABLE 은 오손읽기, 반복할 수 없는 읽기, 팬텀을 초래하는 위반들을 허용하지 않음을 의미함

- 만약, SERIALIZABLE 보다 낮은 고립성 등급으로 실행한다면 아래의 세가지 위반 중 하나 이상이 발생할 수 있음
  - 오손읽기 (Dirty Read)
  - 반복할 수 없는 읽기 (Nonrepeatable Read)
  - 팬텀 현상 (Phantom)

03 SQL의 트랜잭션 지원

2 SQL 트랜잭션의 속성

- 오손읽기
  : 아직 완료되지 않은 T2 가 수정한 항목을 T1이 읽는다고 하면, 만일 T2 가 실패해서 철회된다면 T1 은 존재하지 않는 값을 읽은 꼴이 되어 부정확하게 됨

- 반복할 수 없는 읽기
  : T1 이 테이블로부터 어떤 값을 읽는다고 하면, 만약 이후에 다른 트랜잭션 T2 가 그 값을 수정하고 T1 이 다시 그 값을 읽는다면 T1 은 다른 값을 읽게 될 것임

03 SQL의 트랜잭션 지원

2 SQL 트랜잭션의 속성

팬텀 현상
: T1 이 WHERE 절에 명시된 조건을 근거로 해서 테이블로부터 행들의 집합을 읽는다고 하면, 이제 T2 가 T1 에서 사용되었던 WHERE 절을 만족하는 새로운 한 행을 T1 이 사용했던 테이블에 삽입한다고 하자, 이후 T1 이 반복되면 T1 은 팬텀, 즉 이전에는 존재하지 않았던 하나의 행을 보게 될 것임

03 SQL의 트랜잭션 지원

2 SQL 트랜잭션의 속성

🔎 상이한 고립성 등급에 대한 가능한 위반 요약

	Type of Violation		
Isolation Level	**Dirty Read**	**Nonrepeatable Read**	**Phantom**
READ UNCOMMITTED	Yes	Yes	Yes
READ COMMITTED	No	Yes	Yes
REPEATABLE READ	No	No	Yes
SERIALIZABLE	No	No	No

※ 출처 : 데이터베이스 시스템 6판, Elmasri, Navathe 저, 황규영 외 역, 홍릉과학출판사, 2016년