suanlab

데이터베이스
Database

ORACLE®

DATABASE

12
데이터 무결성과 트랜잭션

NOT NULL
CHECK
COMMIT
ROLLBACK

데이터 무결성과 제약 조건

- 데이터 무결성(Data Integrity)
  - 데이터의 정확성, 유효성, 일관성을 유지하고 보증
  - 데이터베이스나 RDBMS 시스템의 중요한 기능
  - 데이터의 잘못된 입력, 수정, 삭제로부터 보호
  - 입력한 데이터와 데이터베이스에 저장된 데이터 일치

- 데이터 무결성의 종류

유형	설명
개체 무결성 (Entity Integrity)	▪ 기본키(Primary Key) 열은 고유 ▪ Null 값을 가질 수 없음
참조 무결성 (Reference Integrity)	▪ 외래키(Foreign Key)가 있는 테이블은 기본키와의 관계 유지 ▪ 참조하는 외래키가 존재하면 기본키 변경이나 행 삭제 불가
영역 무결성 (Domain Integrity)	▪ 데이터 형태, 범위, 기본값, 유일성에 대한 제한 ▪ 주어진 속성 값은 정의된 도메인에 속한 값
비즈니스 무결성 (Business Integrity)	▪ 사용자의 업무의 특성과 규칙에 따른 제약 조건 ▪ 제약 조건, `DEFAULT`, `TRIGGER` 등의 사용자 정의

- 제약 조건(Constraint)
  - 정해진 규칙에 적합한 데이터만 입력 가능하고, 그 외에는 거부하여 무결성 유지

- 제약 조건의 종류

유형	설명
기본 키 제약 조건	▪ `UNIQUE`와 `NOT NULL` 조건 만족 ▪ 테이블에서 각 행을 유일하게 식별하는 값
외래 키 제약 조건	▪ 외래 키가 참조 열의 값을 반드시 참조해야 함 ▪ 참조되는 열은 `UNIQUE`하거나 기본키
유일 키 제약 조건	▪ 중복된 값을 허용하지 않음 ▪ 유일한 값으로 존재 (null 값 허용 가능)
`NOT NULL` 제약 조건	▪ Null 값을 허용하지 않음 ▪ 값을 반드시 입력해야 함
`CHECK` 제약 조건	▪ 범위나 조건 등 지정된 값만 허용

기본 키 제약 조건

- 기본 키 제약 조건 위반

```
INSERT INTO regions
VALUES (3, 'Asia');
```

```
INSERT INTO countries
VALUES ('AR', 'Argentina', 2);
```

```
INSERT INTO locations
VALUES (1000, 'Street', 12345, 'Korea', null, 'KR');
```

```
INSERT INTO departments
VALUES (10, 'Admin', 200, 1700);
```

```
INSERT INTO employees
VALUES (100, 'Suan' , 'Lee', 'SUAN', '515.123.4567', '21/01/01',
'IT_PROG', 10000, null, null, null);
```

외래 키 제약 조건

- 외래 키 제약 조건 위반

```
INSERT INTO countries
VALUES ('KR', 'South Korea', 5);
```

```
INSERT INTO locations
VALUES (3300, 'Street', 12345, 'Seoul', null, 'KR');
```

```
INSERT INTO departments
VALUES (280, 'Testing', null, 3300);
```

```
INSERT INTO employees
VALUES (207, 'Suan' , 'Lee', 'SUAN', '010.123.1234', '21/01/01',
'IT_QA', 10000, null, null, null);
```

```
INSERT INTO job_history
VALUES (300, '21/01/01', '21/10/01', 'IT_PROG', 300);
```

유일 키, NOT NULL, CHECK 제약 조건

- 유일 키 제약 조건 위반

```
INSERT INTO employees
VALUES (207, 'Suan' , 'Lee', 'SKING', '515.123.4567', '21/01/01',
'IT_PROG', 10000, null, null, null);
```

- NOT NULL 제약 조건 위반

```
INSERT INTO locations
VALUES (3300, 'Street', 12345, null, null, 'US');
```

```
INSERT INTO departments
VALUES (280, null, null, 3300);
```

```
INSERT INTO employees
VALUES (207, 'Suan' , 'Lee', null, '123.123.1234', '21/01/01',
'IT_PROG', 10000, null, null, null);
```

```
INSERT INTO job_history
VALUES (200, null, '21/10/01', 'IT_PROG', 200);
```

```
INSERT INTO job_history
VALUES (200, '21/01/01', null, 'IT_PROG', 200);
```

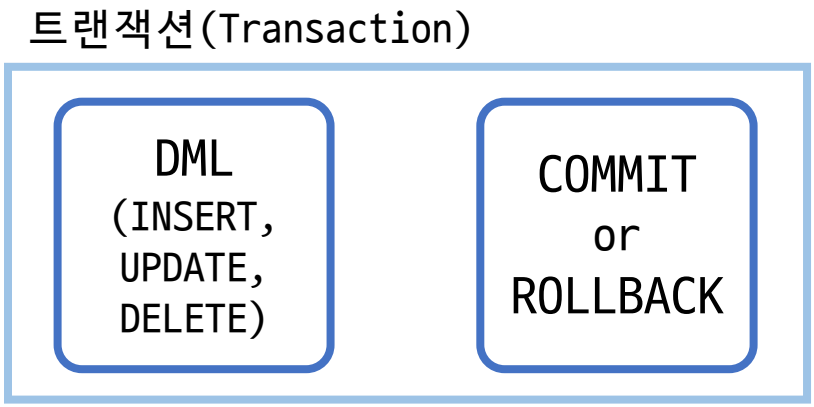
- CHECK 제약 조건 위반

```
INSERT INTO employees
VALUES (207, 'Suan' , 'Lee', 'SUAN', '123.123.1234', '21/01/01',
'IT_PROG', 0, null, null, null);
```

```
INSERT INTO job_history
VALUES (200, '21/10/01', '21/01/01', 'IT_PROG', 200);
```

트랜잭션(Transaction)

- 데이터 무결성이 보장되는 상태에서 데이터 조작 언어(DML, Data Manipulation Language) 작업을 완수하기 위한 논리적인 작업 단위
- DML 실행과 동시성 제어를 위한 중요한 개념
- 데이터 처리시 정상 종료, 사용자 프로세스 실패, 시스템 실패와 같은 비정상 종료에 대한 데이터 신뢰성과 일관성 보장
- DML 작업인 삽입(INSERT), 수정(UPDATE), 삭제(DELETE)를 실행하여 COMMIT 또는 ROLLBACK을 실행하는 과정까지를 트랜잭션이라 부름

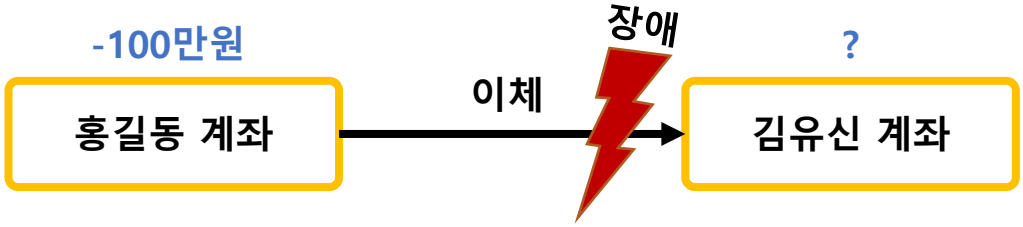


- 트랜잭션의 특징

유형	설명
원자성(Atomicity)	▪ 트랜잭션의 처리가 완전히 끝나지 않았을 경우, 전혀 실행되지 않은 것과 같아야 함(all or nothing)
일관성(Consistency)	▪ 트랜잭션 실행이 성공적으로 완료되면, 데이터베이스는 모순 없이 일관성이 보존된 상태여야 함
고립성(Isolation)	▪ 어떤 트랜잭션도 다른 트랜잭션의 부분적 실행 결과를 볼 수 없음
지속성(Durability)	▪ 트랜잭션이 성공하면 트랜잭션의 결과를 영구적으로 보장해야 함

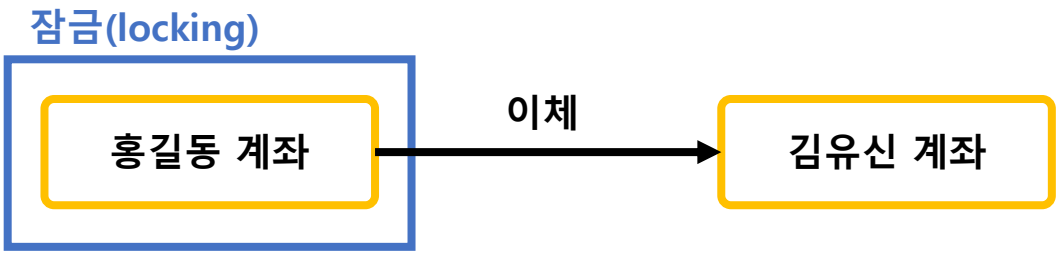
트랜잭션 예제

- 원자성(Atomicity)
  - 계좌 이체 시에 전체 금액이 완전히 이체되거나 전혀 이체되지 않아야 함

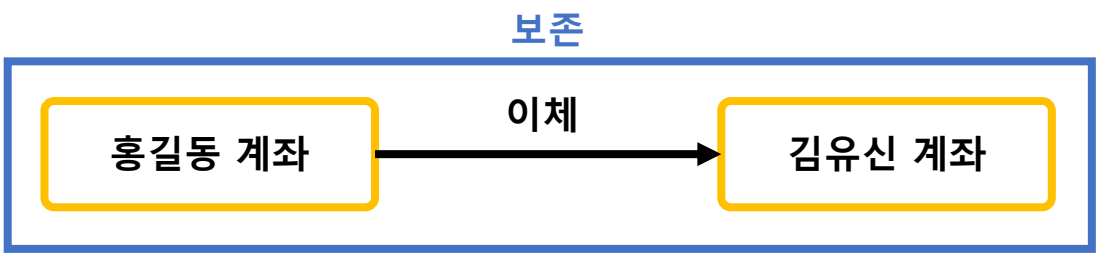


- 일관성(Consistency)
  - 트랜잭션 완료 후에 데이터는 일관되게 유지되어야 함

- 고립성(Isolation)
  - 트랜잭션 완료 전에는 다른 트랜잭션이 참조하거나 변경 불가

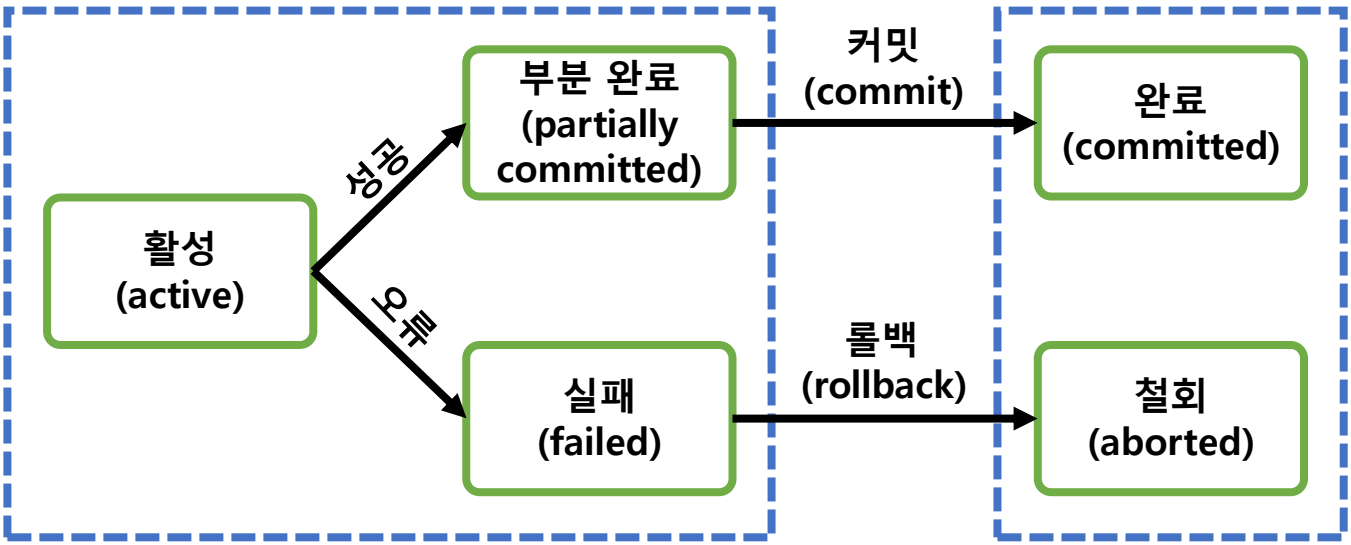


- 지속성(Durability)
  - 트랜잭션 완료 후에 데이터는 저장되어 지속적으로 보존
  - 데이터베이스의 신뢰성과 일관성 유지
  - 장애 발생시에도 복구 가능



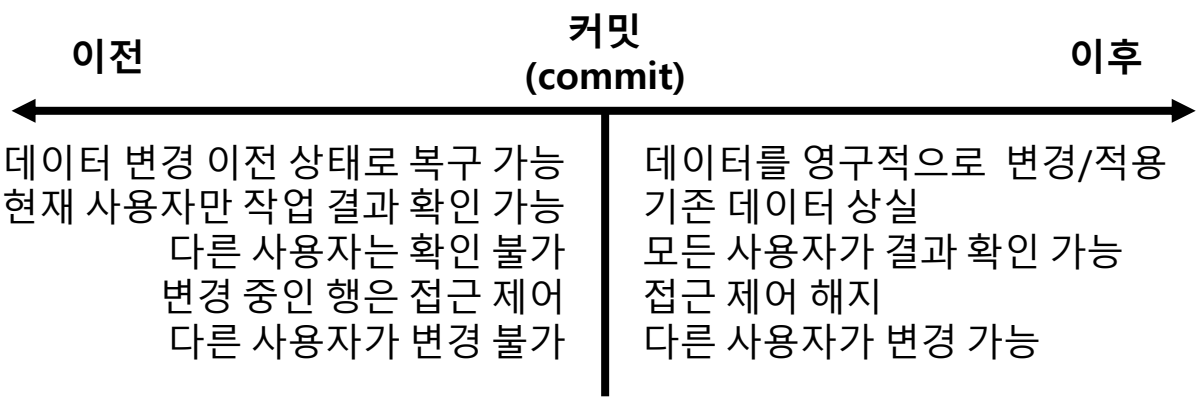
트랜잭션 상태 변화

- 트랜잭션의 수행 단계별 상태 변화 과정



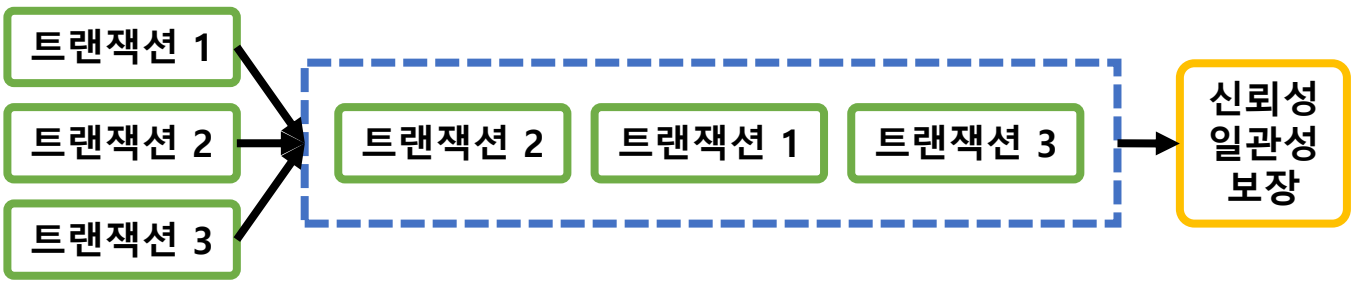
- 활성(active): 트랜잭션 실행 중
- 부분 완료(partially committed): 트랜잭션 명령 실행 후 상태
- 완료(committed): 트랜잭션 성공 완료
- 실패(failed): 더 이상 정상적인 실행 불가
- 철회(aborted): 트랜잭션 복원으로 수행 이전 상태로 복귀

- 커밋(commit): 트랜잭션의 모든 데이터를 영구적으로 반영
- 롤백(rollback): 트랜잭션의 모든 데이터 변경을 포기



동시성 제어(Concurrency Control)

- 다중 사용자(multi-user) 환경을 지원하는 데이터베이스 시스템에서 여러 트랜잭션들이 성공적으로 동시에 실행될 수 있도록 지원
- 다중 사용자 환경을 지원하는 데이터베이스 시스템의 경우 필수적으로 지원해야 하는 기능 (병행 제어)
- 트랜잭션의 직렬화 수행 보장



- 정확한 접근 제어가 안되면 부정확한 데이터 발생
- 동시성 제어 실패 시 데이터 불일치, 갱신 손실 등의 오류 발생

구분	설명
갱신 손실 (Lost Update)	▪ 하나의 트랜잭션이 갱신한 내용을 다른 트랜잭션이 덮어쓰게 되어 갱신이 무효화 ▪ 두 개 이상의 트랜잭션이 한 개의 데이터를 동시에 갱신(update) 할 때 발생
현황 파악 오류 (Dirty Read)	▪ 읽기 작업을 하는 트랜잭션 1이 쓰기 작업을 하는 트랜잭션 2가 작업한 중간 데이터를 읽기 때문에 발생 ▪ 작업중인 트랜잭션 2가 작업을 롤백한 경우 트랜잭션 1은 무효가 된 데이터를 읽어 잘못된 결과 도출
모순성 (Inconsistency)	▪ 다른 트랜잭션들이 해당 항목 값을 갱신하는 동안 한 트랜잭션이 두 개의 항목 값 중 어떤 것은 갱신되기 전의 값을 읽고 다른 것은 갱신된 후의 값을 읽게 되어 데이터 불일치 발생
연쇄 복귀 (Cascading Rollback)	▪ 두 트랜잭션이 동일한 데이터 내용에 접근할 때 발생 ▪ 한 트랜잭션이 데이터를 갱신한 다음 실패하여 롤백 연산을 수행하는 과정에서 갱신과 롤백 연산을 실행하고 있는 사이에 해당 데이터를 읽어서 사용할 때 발생

동시성 제어(Concurrency Control)

- 동시성 제어 기법

제어 기법	설명
Locking (Shared Lock)	▪ 데이터 항목에 대해 읽기(read)만 가능
Locking (Exclusive Lock)	▪ 데이터 항목에 대해 읽기와 기록(입력/삭제) 모두 불가능
2 Phase Locking	▪ 모든 트랜잭션들이 lock과 unlock 연산을 확장 단계와 수축 단계로 구분하여 수행
Timestamp Ordering	▪ 데이터베이스 시스템에 들어오는 트랜잭션 순서대로 System Clock / Logical Counter 할당하고 순서를 부여하여 동시성 제어의 기준으로 사용
Validation	▪ 트랜잭션 수행 동안은 어떠한 검사도 하지 않고, 트랜잭션 종료 시 일괄적 검사 기법
MVCC (Multi-Version Concurrency Control)	▪ 트랜잭션의 타임스탬프와 접근 데이터의 여러 버전 타임스탬프를 비교하여 직렬 가능성이 보장되는 버전 선택

- 동시성 제어 기법 비교

제어 기법	장점	단점
2 Phase Locking	▪ 데이터 오류 가능성 예방 ▪ 간단한 알고리즘	▪ Lock 대기시간 발생 ▪ Deadlock 발생
Timestamp Ordering	▪ Deadlock 발생 없음 ▪ 트랜잭션 대기 시간 없음	▪ Rollback 발생 확률 높음 ▪ Cascading Rollback 가능
Validation	▪ 동시 처리 능력 증가 ▪ 트랜잭션 대기 시간 없음	▪ 장기 트랜잭션 철회 시 자원 낭비
MVCC (Multi-Version Concurrency Control)	▪ 최근 데이터 값 선택 ▪ 동시성, 일관성 동시 해결	▪ Undo 블록 I/O에 따른 오버헤드 발생


suanlab
이수안 컴퓨터 연구소
suan computer laboratory