

트랜잭션 격리 수준

Transaction Isolation Level

2025.07.30.

System Software Lab.

목차

01 READ UNCOMMITTED

02 READ COMMITTED

03 REPEATABLE READ

04 SERIALIZABLE

트랜잭션 격리성

- 각각의 트랜잭션은 서로 간섭 없이 독립적으로 수행되어야 한다.

트랜잭션이 끼어들지 못하면...?

→ 트랜잭션이 순차적으로 처리

→ 처리 대기 중인 트랜잭션이 늘어남

→ 평균 트랜잭션 처리 시간이 늘어남

- 트랜잭션 처리 속도와 데이터의 일관성 간 적절한 조율(트레이드 오프)이 필요

→ 단계별로 조율한 것이 트랜잭션 격리 수준

- 1992년 SQL 표준인 SQL-92에서 트랜잭션 격리 수준이라는 개념을 도입 미국 카네기 멜런 대학교의 앤드류 프로젝트 홈페이지에 표준을 찾을 수 있음
- MySQL에서는 전역(모든 세션) 또는 세션 단위로 트랜잭션 격리 수준을 설정할 수 있음
 - READ UNCOMMITTED
 - READ COMMITTED
 - REPEATABLE READ
 - SERIALIZABLE

READ UNCOMMITTED

- 다른 트랜잭션에서 커밋되지 않은 데이터에 접근할 수 있게 하는 격리 수준
- 각 트랜잭션에서의 변경 내용이 **COMMIT**이나 **ROLLBACK** 여부에 상관 없이 다른 트랜잭션에서 접근 가능
- 트랜잭션에 문제가 발생하여 **ROLLBACK**이 수행되면 트랜잭션 수행 중 다른 트랜잭션에서 읽은 데이터가 변경되므로 부정합 발생
- Oracle Database, PostgreSQL 등 일부 RDBMS에서는 지원하지 않음

READ UNCOMMITTED

- 다른 트랜잭션에서 커밋되지 않은 데이터에 접근할 수 있게 하는 격리 수준
- 각 트랜잭션에서의 변경 내용이 **COMMIT**이나 **ROLLBACK** 여부에 상관 없이 다른 트랜잭션에서 접근 가능

id	name
1	우정잉
2	도유정

START TRANSACTION;

1	우정잉
2	도유정

UPDATE;

COMMIT;

1	우정잉
2	김우정

START TRANSACTION;

1	우정잉
2	김우정

COMMIT;

1	우정잉
2	김우정

The screenshot displays a database IDE with two active console windows, 'console' and 'console_2', both connected to a 'local-mysql' instance. The 'console' window contains the following SQL statements:

```
1 START TRANSACTION;
2 -- user 테이블을 조회한다
3 SELECT * FROM user;
4 -- (2, 도유정)을 (2, 김우정)으로 바꾼다
5 UPDATE user SET name = '김우정' WHERE id = 2;
6 COMMIT;
```

The 'console_2' window contains the following SQL statements:

```
1 START TRANSACTION;
2 SELECT * FROM user;
3 COMMIT;
```

The 'Services' panel at the bottom shows the execution timeline of these transactions:

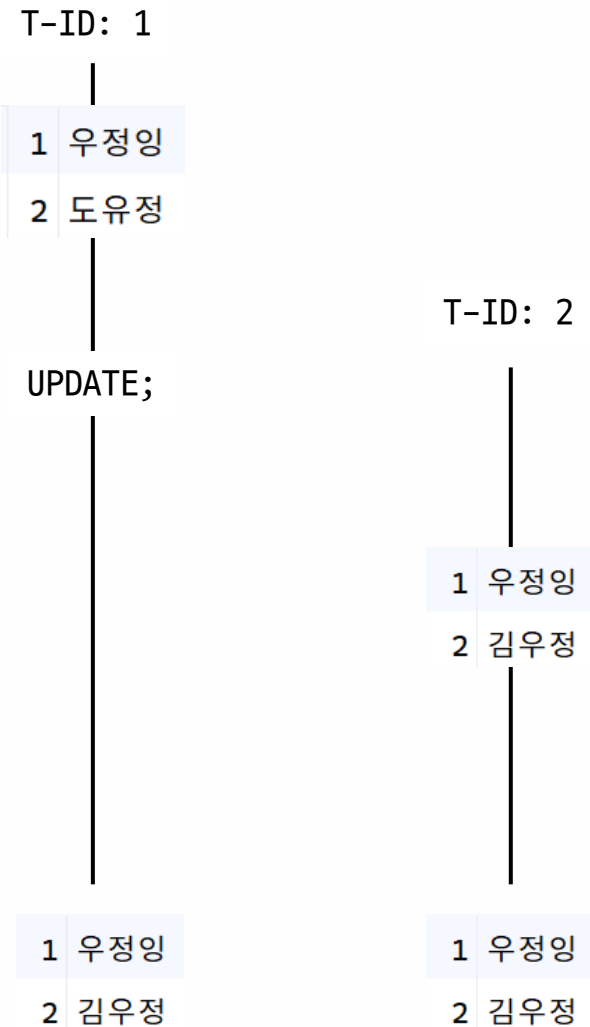
- local-mysql@localhost: default 75 ms
- console: 70 ms
- console_2: 174 ms

The 'Output' window at the bottom right shows the result of the SELECT query in the 'console' window, displaying two rows of data:

id	name
1	우정잉
2	도유정

The status bar at the bottom indicates the current settings: 6:8, CRLF, UTF-8, 4 spaces.

Dirty Read



- 다른 트랜잭션에서 커밋되지 않은 데이터에 접근하여 부정합을 일으킬 수 있는 데이터를 읽을 수 있는 문제
- 개명 반영 중 오류가 발생했다면?
- **주민등록 API 사용자가 잘못된 이름을 인식하게 됨**
- 가 군이 나 양에게 만 원 송금 중 나 양이 계좌 잔액을 조회할 때 서버 장애로 가 군의 송금이 실패했다면?
- **나 양이 가 군이 송금했다고 판단할 수 있음**

READ COMMITTED

- 다른 트랜잭션에서 커밋된 데이터에만 접근할 수 있게 하는 격리 수준
- 각 트랜잭션에서의 변경 내용이 **COMMIT**이나 **ROLLBACK**되어야 다른 트랜잭션에서 접근할 수 있음
- MySQL의 Undo Tablespace을 참조해 변경 이전의 내용을 다른 트랜잭션이 조회

id	name
1	우정잉
2	김우정

START TRANSACTION;

1 우정잉
2 김우정

UPDATE;

1 우정잉
2 도유정

START TRANSACTION;

1 우정잉
2 김우정

1 우정잉
2 도유정

1 우정잉
2 도유정

The screenshot shows a database console with two active transactions, console_3 and console_4, connected to a local-mysql instance.

console_3:

```
1 START TRANSACTION;
2 -- user 테이블을 조회한다
3 SELECT * FROM user;
4 -- (2, 김우정)을 (2, 도유정)으로 바꾼다
5 UPDATE user SET name = '도유정' WHERE id = 2;
6 COMMIT;
```

console_4:

```
1 START TRANSACTION;
2 SELECT * FROM user;
3 -- console_3의 트랜잭션이 끝난 이후 실행됨
4 SELECT * FROM user;
5 COMMIT;
```

Services:

- Database
 - local-mysql@localhost
 - default 75 ms
 - console_3 42 ms
 - console_3 42 ms
 - console_4 38 ms
 - console_4 38 ms

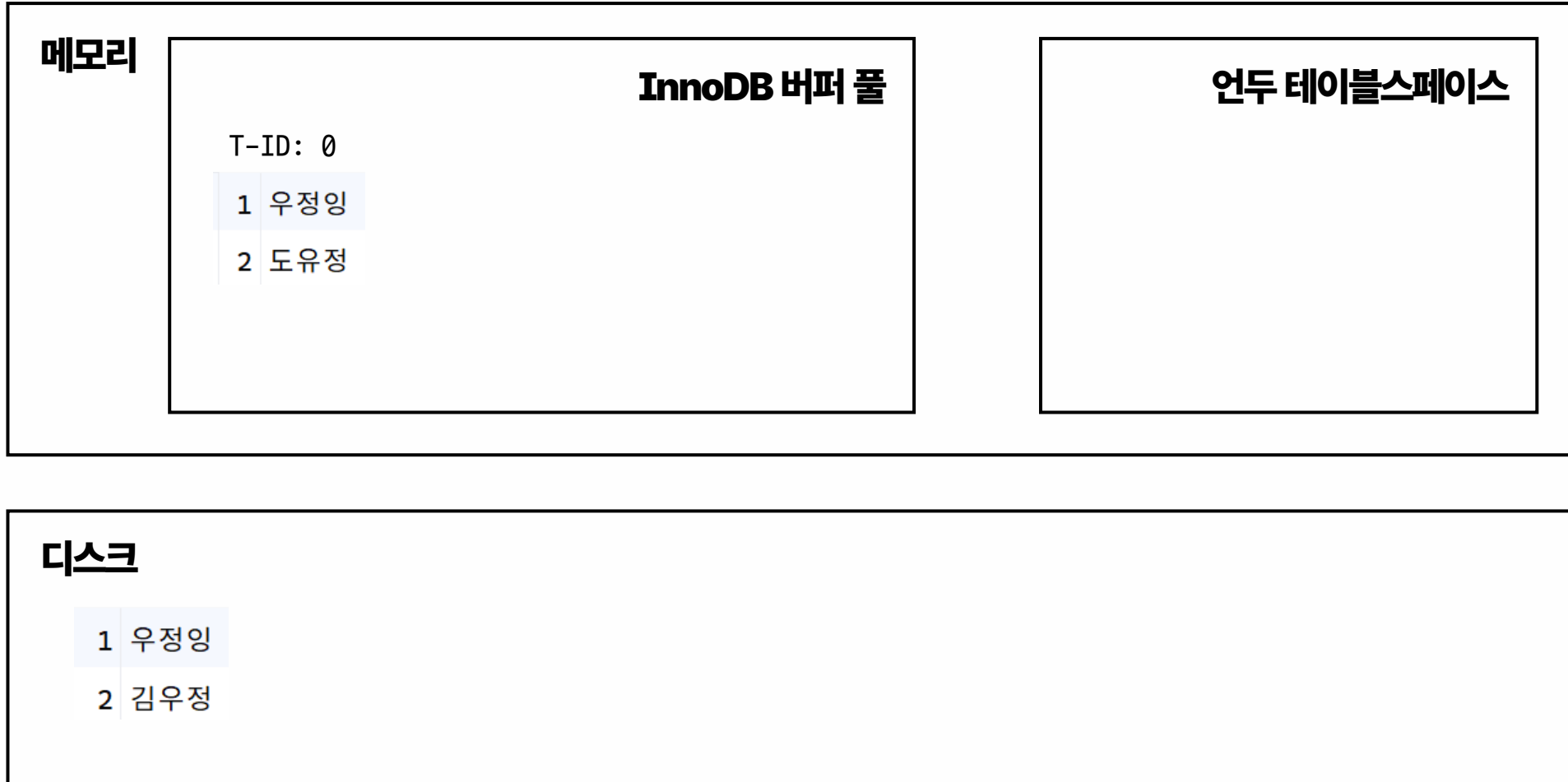
Output: console_3의 트랜잭션이 끝난 이후 실행됨

id	name
1	우정잉
2	김우정

2 rows

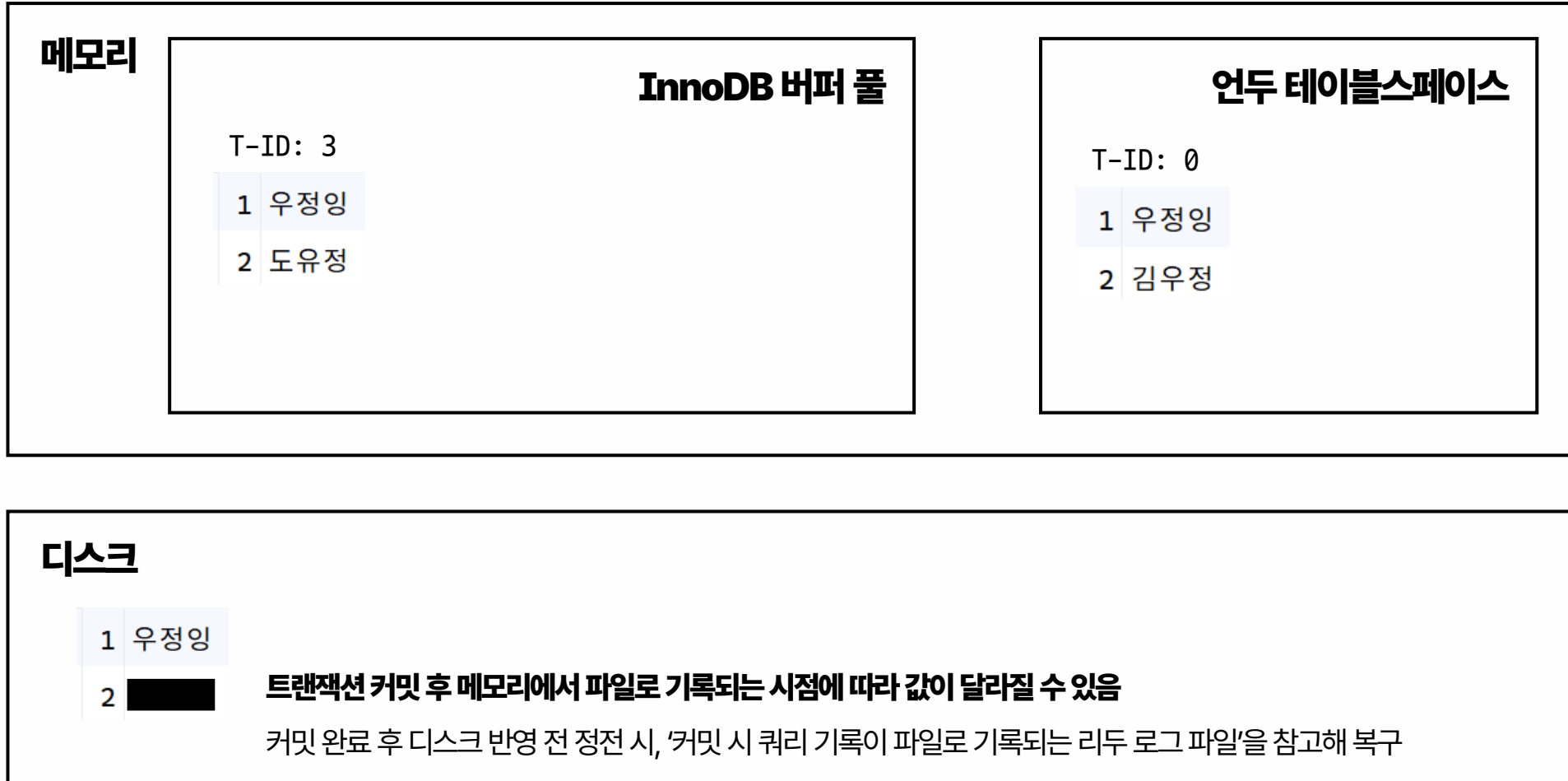
Undo Tablespace

- 트랜잭션 ROLLBACK 지원과 일관된 읽기를 제공하기 위해 변경 전 데이터를 저장

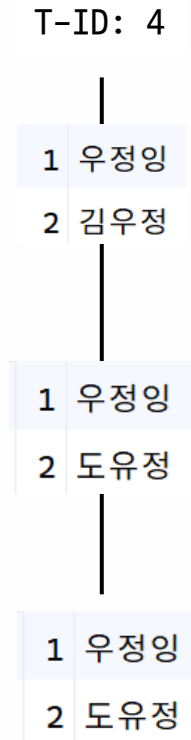


Undo Tablespace

- 트랜잭션 ROLLBACK 지원과 일관된 읽기를 제공하기 위해 변경 전 데이터를 저장



Non Repeatable Read



- 같은 트랜잭션에서 동일한 조회 쿼리를 실행했음에도 다른 결과가 나타나는 문제
- 개명 중 주민등록 API를 호출했다면?
- **개명 중과 개명 후 이름 정보가 달라 검증 실패가 발생할 수 있음**
- 가 군이 나 양에게 만 원 송금 중 자동 이체가 이루어져야 한다면?
- **자동 이체 트랜잭션이 잘못된 금액을 확인하고 계좌 잔고를 갱신할 수 있음**

REPEATABLE READ

- 다른 트랜잭션에서 커밋된 데이터를 읽을 수 있되
자신보다 낮은 트랜잭션 번호를 갖는 트랜잭션에서 커밋한 데이터만 읽게 하는 격리 수준
- 낮은 번호를 갖는 트랜잭션에서의 변경 내용이 **COMMIT**이나 **ROLLBACK**되어야 다른 트랜잭션에서 접근할 수 있음
- Undo Tablespace에 여러 버전의 정보가 저장되므로 *(다중 버전 동시성 제어)* 가능한 격리 수준

REPEATABLE READ

- 다른 트랜잭션에서 커밋된 데이터를 읽을 수 있되
자신보다 낮은 트랜잭션 번호를 갖는 트랜잭션에서 커밋한 데이터만 읽게 하는 격리 수준

id	name
1	우정잉
2	도유정

START TRANSACTION;

1 우정잉
2 도유정

UPDATE;

1 우정잉
2 김우정

START TRANSACTION;

1 우정잉
2 도유정

1 우정잉
2 도유정

1 우정잉
2 김우정

```
console x
1 START TRANSACTION;
2 -- user 테이블을 조회한다
3 SELECT * FROM user;
4 -- (3, 도유정)을 추가한다
5 INSERT INTO user (id, name) VALUES (id 3, name '도유정');
6 COMMIT;
7

console_5 x
1 START TRANSACTION;
2 SELECT * FROM user;
3 SELECT * FROM user;
4 COMMIT;
```

Services

- Database
 - local-mysql@localhost
 - console_5
 - console_5
 - console
 - console

Information

Dialect: MySQL

DBMS: MySQL (ver. 9.3.0)

Case sensitivity: plain=mixed, delimited=mixed

Driver: MySQL Connector/J (ver. mysql-connector-j-8.2.0 (Revision: 06a1f724497fd81c6a659131fda822c9e5085b6c), JDBC4.2)

Phantom Read



- 같은 트랜잭션에서 동일한 개수 조회 쿼리를 실행했음에도 다른 결과가 나타나는 문제
- 개명 중 주민등록 API를 호출했다면?
- **개명 중과 개명 후 같은 이름을 가진 사람의 수가 다르게 표시될 수 있음**
- Oracle Database 등에서는 Phantom Read가 발생할 수 있어 격리 수준을 고수준(Serializable)으로 높여야 하지만,
- PostgreSQL, MySQL, MariaDB에서는 스냅샷(*Undo Tablespace*)을 만들어 트랜잭션 시작 시점의 데이터를 조회할 수 있게 함
- 그러나, 잠금을 건 읽기 작업의 경우 스냅샷이 아닌 실제 테이블 상태를 참조하여 Phantom Read가 발생할 수 있음

잠금 읽기를 통한 Phantom Read 확인

- 같은 트랜잭션에서 동일한 개수 조회 쿼리를 실행했음에도 다른 결과가 나타나는 문제

The screenshot displays a database IDE with two active SQL console windows, 'console' and 'console_5', both connected to a local MySQL instance. The 'console' window contains the following SQL statements:

```
1 START TRANSACTION;
2 -- user 테이블을 조회한다
3 SELECT * FROM user;
4 -- (3, 도유정)을 추가한다
5 INSERT INTO user (id, name) VALUES (id 3, name '도유정');
6 COMMIT;
```

The 'console_5' window contains the following SQL statements:

```
1 START TRANSACTION;
2 SELECT * FROM user;
3 SELECT * FROM user for update;
4 COMMIT;
```

Below the console windows, the 'Services' panel shows a tree view of the database connections. Under 'local-mysql@localhost', there are two active transactions: 'console_5' (75 ms) and 'console' (72 ms). The 'Information' panel on the right provides details about the MySQL environment:

- Dialect: MySQL
- DBMS: MySQL (ver. 9.3.0)
- Case sensitivity: plain=mixed, delimited=mixed
- Driver: MySQL Connector/J (ver. mysql-connector-j-8.2.0 (Revision: 06a1f724497fd81c6a659131fda822c9e5085b6c), JDBC4.2)

The status bar at the bottom indicates the current database is 'local-mysql@localhost' and shows settings like '3:20', 'CRLF', 'UTF-8', and '4 spaces'.

SERIALIZABLE

- 트랜잭션을 무조건 순차적으로 실행하여 데이터 일관성은 유지되나 동시 처리가 불가능해 처리 속도가 느려짐
- 데이터 조회 시 다른 트랜잭션이 변경(삽입, 갱신, 삭제) 쿼리를 실행할 수 없게 하며
데이터 변경 시에는 읽기, 변경 쿼리를 실행할 수 없게 함

T-ID: 7
|
1 | 우정잉
2 | 김우정
|
UPDATE;
|
1 | 우정잉
2 | 도유정
|

T-ID: 8
|
1 | 우정잉
2 | 도유정
|
1 | 우정잉
2 | 도유정
|

정리

트랜잭션 성질 중 하나인 격리성

각각의 트랜잭션은 서로 간섭 없이 독립적으로 수행되어야 한다.
한 트랜잭션 내에서 데이터는 동일해야 한다.

격리성을 보장을 위해서는...

트랜잭션 수행 중 다른 트랜잭션이 중간에 끼어들지 못 하게 해야 함

처리 속도가 느려진다!

트랜잭션 처리 속도와 데이터의 일관성 간 적절한 조율(*트레이드 오프*)이 필요

트랜잭션 격리 수준

READ UNCOMMITTED Dirty Read, Non Repeatable Read, Phantom Read

커밋하지 않은 데이터에도 접근할 수 있으며 데이터가 조회되었다 사라질 수 있음

READ COMMITTED Non Repeatable Read, Phantom Read

커밋된 데이터만 조회할 수 있으며 커밋 여부에 따라 데이터가 달라질 수 있음

REPEATABLE READ Phantom Read

한 트랜잭션 내에서 동일한 결과를 보장하지만 새로운 레코드가 추가되면 부정합 발생 가능

SERIALIZABLE

가장 엄격한 수준의 격리 수준으로, 트랜잭션을 순차적으로 실행하여 안전하지만 성능이 떨어짐

들어 주셔서 감사합니다

E-mail	park@duck.com
LinkedIn	yeonjong-park
Instagram	yeonjong.park
GitHub	patulus

2025.07.30.

System Software Lab.