Lecture 15: database transaction

BADM/ACCY 352

Spring 2017

Instructor: Yi Yang, PhD

- Class project milestone 2 due today

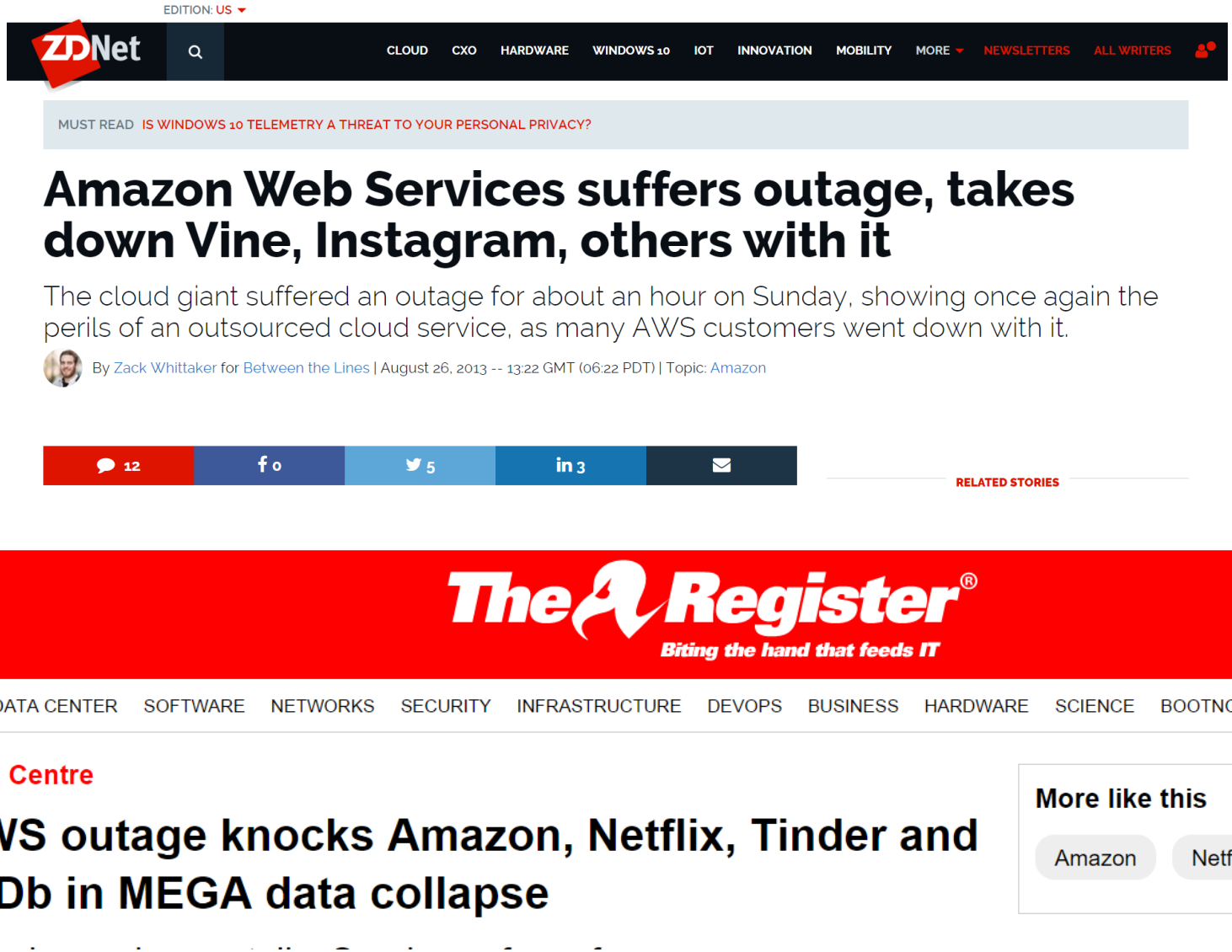
- On Wednesday, guest speaker
- Shaun Thomas, Database Architect at PEAK6 Investments

Previous Lecture

- Database normalization
  - How to convert an un-normalized table to 1NF, 2NF, 3NF

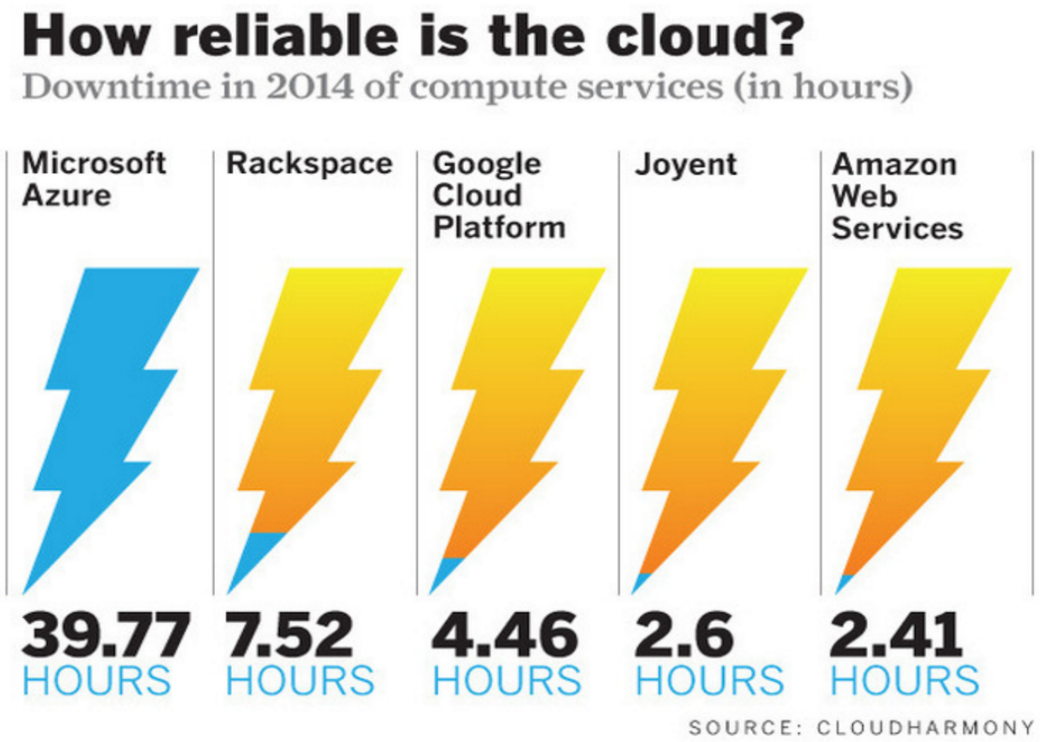
This Lecture

- We will learn transaction and concurrency control in modern database systems.



What does it mean that a website is down?

- Service Level Agreement (SLA):
  - Amazon Web Service: 99.95% ~4hours in a year
  - Microsoft Azure
    - 99.9% for basic tier, ~ 9 hours in a year
    - 99.99% for premium tier ~ less than a hour in a year



Not only cloud

BBC | Sign in | News | Sport | Weather | Shop | Earth | Trave

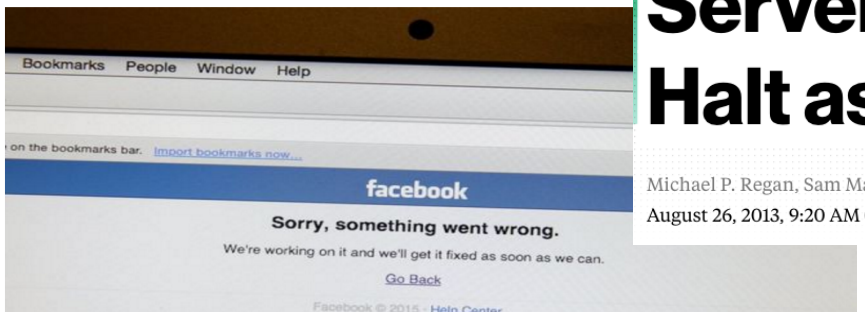
NEWS

Home | Video | World | US & Canada | UK | Business | Tech | Science | Magazine | Ente

US & Canada

Facebook down for second time in a week

28 September 2015 | US & Canada



Server Crash Spurs 3-Hour Nasdaq Halt as Data Link Lost

Michael P. Regan, Sam Mamudi and Whitney Kisling
August 26, 2013, 9:20 AM CDT

JPMorgan Chase Suffers Online Banking Outage, Confirms Cyberattack

By Brian Browdie
March 12, 2013

Twitter | LinkedIn | Facebook | Google + | Email | Comments | Print | Reprints

Reasons

- Software Error/Bug
- Cyber attack
- Hardware Failure
  - Power outage due to extreme weather

ASIA-PACIFIC, COOLING, DOWNTIME

Heatwave, Cooling Failure Bring iiNet Data Center Down in Perth

BY JASON VERGE ON JANUARY 6, 2015 ADD YOUR COMMENTS

36 Like | 72 Tweet | 77 Share | 9 G+1

Hurricane Sandy takes data centers offline with flooding, power outages

Hosting customers stranded as generators in NY data centers run out of fuel.

by **Jon Brodkin** - Oct 30, 2012 11:25am CDT

Share Tweet 100

So…

Let's say you transfer money from one bank account to another.

1. withdraw the amount from the source account (update database)

oops, Outage happens!

2. deposit it to the destination account (update database)

Database as a data store treats the integrity of data as paramount.

1. Relational modeling + normalization

2. Database transaction + concurrency control

Database transaction

- A transaction is a logical unit of work that must be entirely completed or entirely aborted, no intermediate states are acceptable.
- A transaction may consist of a single SQL statement or a collection of related SQL statement.

Database transaction

- If any of the SQL statements in the transaction fail, the entire transaction is rolled back to the original database state that existed before the transaction started.
- A successful transaction changes the database from one consistent state to another.
  - Will SELECT command change system state?
- All or nothing.

Transaction in SQL

- SQL standard defines two commands to control transaction
  - COMMIT: to save the changes
  - ROLLBACK: to rollback the changes

```
SQL> DELETE FROM CUSTOMERS
     WHERE AGE = 25;
SQL> COMMIT;
```

```
SQL> DELETE FROM CUSTOMERS
     WHERE AGE = 25;
SQL> ROLLBACK;
```

- By default, MySQL commits changes automatically after each command.

student

stu_id	stu_name	stu_dob	stu_gpa
001	alice	01-01-1991	3.6
002	bob	02-01-1991	3.3
003	charlie	03-01-1992	3.8

```
START TRANSACTION;
UPDATE STUDENT SET STU_GPA = 3.3 WHERE STU_ID = 001;
COMMIT;
```

```
START TRANSACTION;
UPDATE STUDENT SET STU_GPA = 3.3 WHERE STU_ID = 001;
ROLLBACK;
```

So…

Let's say you transfer money from one bank account to another.

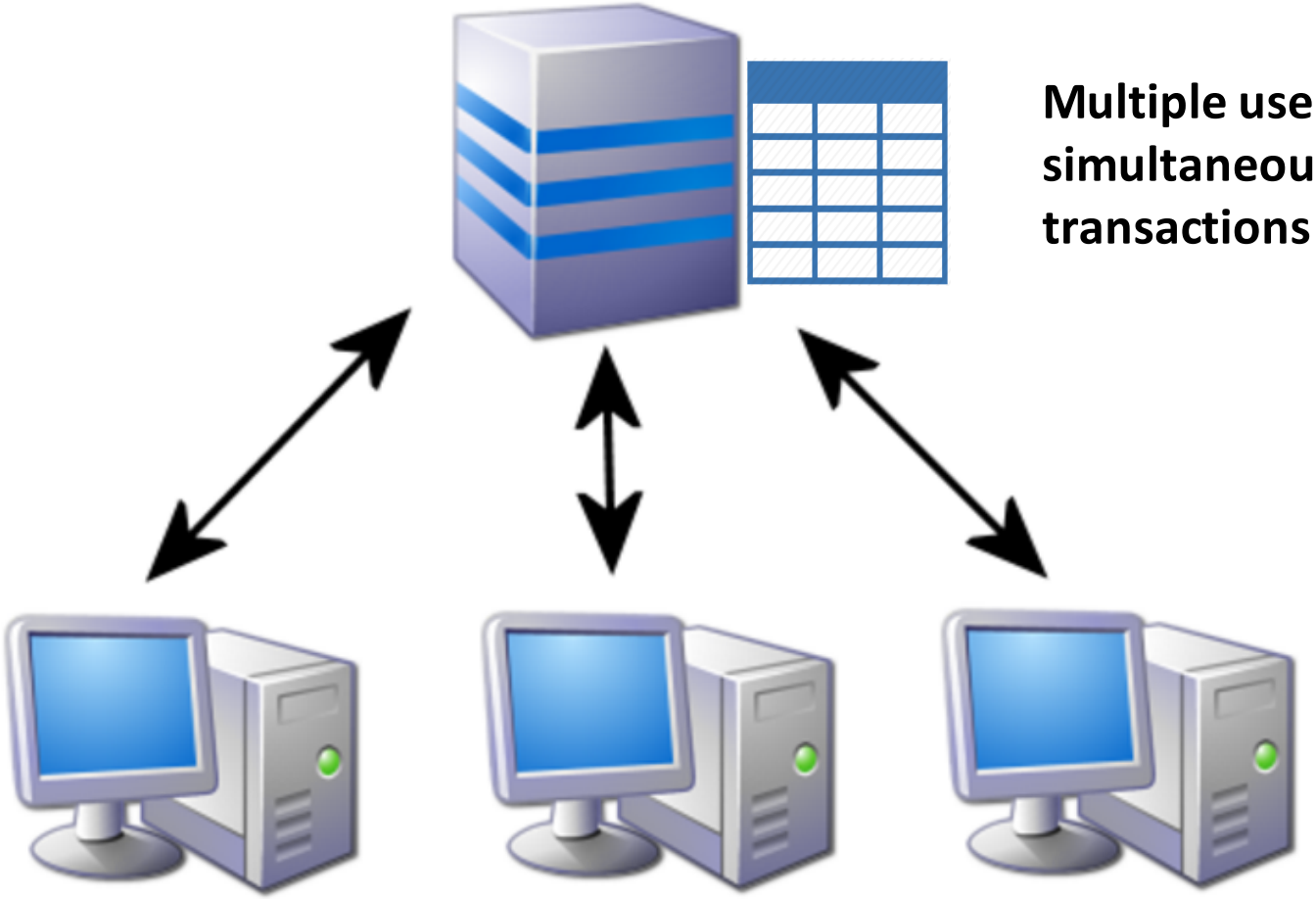
one transaction

1. withdraw the amount from the source account (update database)
2. deposit it to the destination account (update database)

Transaction Properties: ACID

- Atomicity: A transaction is treated as a single logical unit of work.
- Consistency: A transaction takes a database from one consistent state to another.
- Isolation: If transaction T1 is being executed and is using the data item X, X cannot be accessed by any other transactions until T1 ends. (we will see later)
- Durability: once transaction changes are done, it will remain so permanently. Even a system failure.

Concurrency control in Multi-user database



Multiple users will simultaneously execute transactions in a database.

Concurrency problem in multi-user database

Let's say an order table contains 1,000,000 rows, with a disk size of 20 GB.

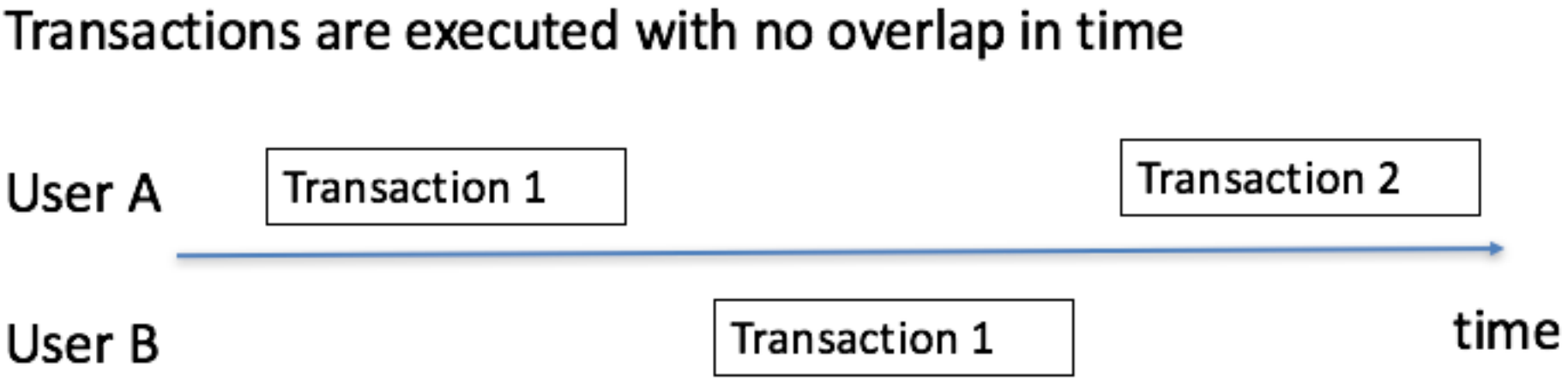
8:00: UserA started a query "SELECT * FROM order". This query takes approximately 5 minutes to complete, as the database must fully scan the table's from start to end.

8:01: UserB deleted the last row in the order table, and commits the change.

8:04: UserA's query arrives at the end of the order table. **What will happen?** Can UserA see the row deleted by UserB?

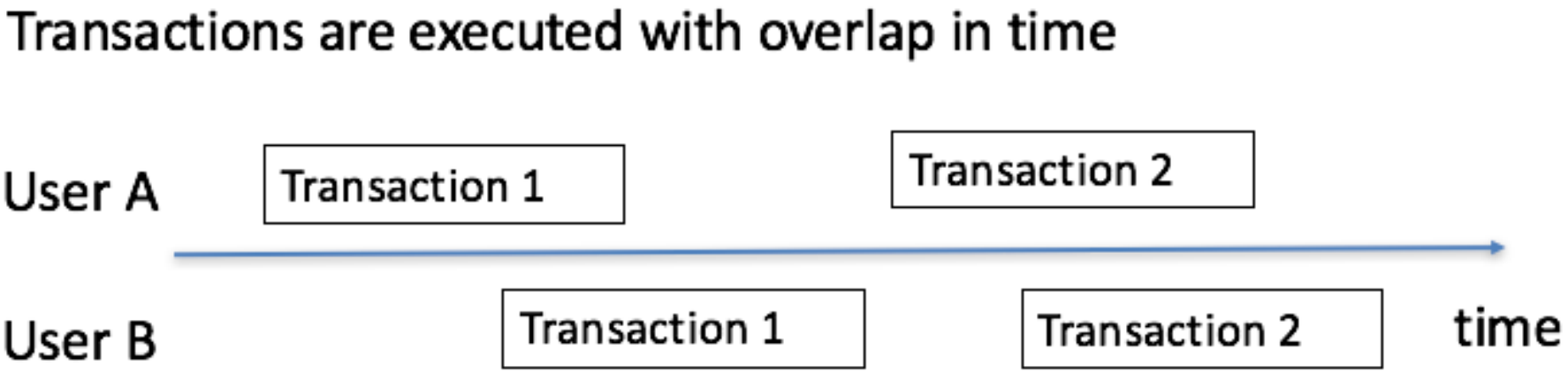
Why is concurrency control needed?

- **Ideally:** If transactions are executed sequentially with no overlap in time, no transaction concurrency exists.



Why is concurrency control needed?

- **In reality:** However, in a multi-user database environment where concurrent transactions with interleaving operations exist, some unexpected, undesirable result may occur.



Dirty Read

T1

```
/* Query 1 */
SELECT age FROM users WHERE id = 1;
/* will read 21 */
```

T2

```
/* Query 1 */
SELECT age FROM users WHERE id = 1;
/* will read 20 */
```

```
/* Query 2 */
UPDATE users SET age = 21 WHERE id = 1;
/* No commit here */
```

```
ROLLBACK; /* lock-based DIRTY READ */
```

T1 read uncommitted data that is updated by T2.
Since there is no guarantee that the data will ever be committed, the data read by T1 is *dirty*.

Non-Repeatable Read

T1

```sql
/* Query 1 */
SELECT * FROM users WHERE id = 1;
```

T2

```sql
/* Query 2 */
UPDATE users SET age = 21 WHERE id = 1;
COMMIT; /* in multiversion concurrency
   control, or lock-based READ COMMITTED */
```

```sql
/* Query 1 */
SELECT * FROM users WHERE id = 1;
COMMIT; /* lock-based REPEATABLE READ */
```

During the course of T1, a row is retrieved twice
BUT the values within the row differ between reads.

Phantom Read

T1

```
/* Query 1 */
SELECT * FROM users
WHERE age BETWEEN 10 AND 30;
```

T2

```
/* Query 2 */
INSERT INTO users(id,name,age) VALUES ( 3, 'Bob', 27 );
COMMIT;
```

```
/* Query 1 */
SELECT * FROM users
WHERE age BETWEEN 10 AND 30;
COMMIT;
```

During the course of T1, two identical queries are executed, BUT the collection of rows returned by the second query is different from the first.

Differences

- Dirty reads happens when T1 reads an un-committed data modified by T2.
- In both non-repeatable read and phantom read, T2 commits the data modification, however, non-repeatable read happens when T2 updates the data, and phantom read happens when new data is inserted/deleted.

Read/Write Conflict

If two transactions are executed simultaneously, will they conflict?

Write: insert/delete/update operation.
Read: select operation

	Transactions		
	T1	**T2**	**Conflict?**
Operations	**Read**	**Read**	**No Conflict**
	Read	**Write**	**Conflict**
	Write	**Read**	**Conflict**
	Write	**Write**	**Conflict**

Concurrency Control

- Coordinating the simultaneous execution of transactions in a multi-user database system is called concurrency control.
- This mechanism enforces **data integrity** and **data consistence**.
- How? Using locks!

Lock

- A lock guarantees *exclusive* use of data to a current transaction.
- T2 has no access to the data that is currently being used by T1.
- Lock granularity
  - Database level
  - Table level
  - Range level
  - Row level

Lock example

stu_id	stu_name	stu_dob	stu_gpa
0001	alice	01-01-1991	3.6
0002	bob	02-01-1991	3.3
0003	charlie	03-01-1992	3.8

Database lock

Table lock

Range lock

Row lock

Question

Given the four levels of lock granularity, which one do you think gives the slowest data access performance?

a. Database level
b. Table level
c. Range level
d. Row level

Which level of lock to use?

- If the database uses table level lock, will dirty read happen?
- If the database uses row level lock, will dirty read happen? What about non-repeatable read? What about phantom read?

Isolation level

		Isolation Level			
		Read Uncommitted	Read Committed	Repeatable Read	Serializable
Problem Type	Dirty Read	Possible	Not Possible	Not Possible	Not Possible
	Nonrepeatable Read	Possible	Possible	Not Possible	Not Possible
	Phantom Read	Possible	Possible	Possible	Not Possible

Less restrictive

More restrictive

MySQL uses Repeatable read as default isolation level.

Transaction Properties: ACID

- Atomicity: A transaction is treated as a single logical unit of work.
- Consistency: A transaction takes a database from one consistent state to another.
- Isolation: If transaction T1 is being executed and is using the data item X, X cannot be accessed by any other transactions until T1 ends. (Different isolation level)
- Durability: once transaction changes are done, it will remain so permanently. Even a system failure.

Summary

- Why transaction is needed in database, and what is it?
- Why concurrency control is needed in database, and what is it?

- Understand ACID property.