Database Concepts

Lecture 1

Introduction

Why do you choose this course?

I heard it's easy.

I heard it's useful.

It fits my schedule.

I took Prof. Deng's CS 262 before.

None of the above

People

- Instructor: Dr. Ping Deng
  - E-mail: pideng@gmu.edu
  - Office hours: MW 12-1 PM @ ENGR 4608 or by appointment
- GTAs:
  - Bobby Chab: rchab@gmu.edu
    - Office hours: TBD
  - Syeda Sabrina Akter: sakter6@gmu.edu
    - Office hours: TBD
- UTAs:
  - Rahima Adnan
  - Bahy Ba Huynh
  - Seung Ju Lee

Resources

- Blackboard for assignments/submissions, lecture slides and pre-recorded lectures
- Piazza for discussions/questions
  - https://piazza.com/gmu/fall2023/cs450001003
  - Honor code applies!
- Textbooks: Fundamentals of Database System (7th Edition) by Ramez Elmasri and Shamkant B. Navathe
- Recommended books:
  - Oracle 10g Programming: A Primer by Sunderraman
  - NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence by Sadalage and Fowler

Grading

- ❑ Class participation: 5%
- ❑ Quizzes: 10%
- ❑ Projects: 35%
- ❑ Midterm Exam: 25%
- ❑ Final Exam: 25%

- The lowest quiz score for the semester will be dropped
- A budget of 3 late days you can use for projects (No late work accepted otherwise)
- Grades will be changed only when a grading error has been made. All grade change requests are due within **a week** of the grade becoming available on Blackboard
- There will be no make-up of exams or quizzes unless **previously** arranged with the instructor
- **Unexcused absence** from the final exam will result in 0 for the final exam
- Extra credit might be available on specific quiz, exam or assignment, but not as an end-of-semester batch of extra work

Policies and Tips

- Honor code applies to all graded work in the class. No AI tools can be used for any graded work unless stated otherwise. All cases will be forwarded to the Honor Committee with the recommendation for an **F** in the course
- You may resubmit assignments before deadline
- No submission may be made after an assignment has been graded
- Time management is essential

Are you graduating this semester?

Topics

- ER & EER model
- Relational data model
- ER & EER to relational mapping
- Relational algebra
- SQL
- Database programming
- Functional dependency and normalization
- NoSQL

What is a Database?

- A collection of related data
  - Data: Known facts that can be recorded and have an implicit meaning
- Mini-world
  - Some part of the real world about which data is stored in a database
- A logically coherent collection of data with some inherent meaning
- Designed, built, and populated with data for a specific purpose

Database Example

- A database that stores student records and their grades

STUDENT	Name	StudentNumber	Class	Major
	Smith	17	1	CS
	Brown	8	2	CS

COURSE	CourseName	CourseNumber	CreditHours	Department
	Intro to Computer Science	CS1310	4	CS
	Data Structures	CS3320	4	CS
	Discrete Mathematics	MATH2410	3	MATH
	Database	CS3380	3	CS

SECTION	SectionIdentifier	CourseNumber	Semester	Year	Instructor
	85	MATH2410	Fall	98	King
	92	CS1310	Fall	98	Anderson
	102	CS3320	Spring	99	Knuth
	112	MATH2410	Fall	99	Chang
	119	CS1310	Fall	99	Anderson
	135	CS3380	Fall	99	Stone

GRADE_REPORT	StudentNumber	SectionIdentifier	Grade
	17	112	B
	17	119	C
	8	85	A
	8	92	A
	8	102	B
	8	135	A

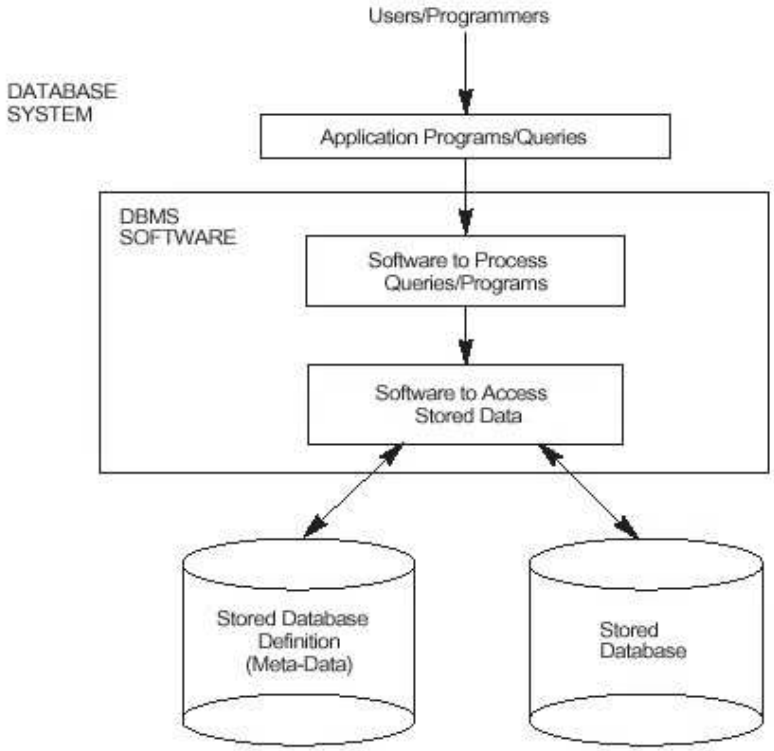
PREREQUISITE	CourseNumber	PrerequisiteNumber
	CS3380	CS3320
	CS3380	MATH2410
	CS3320	CS1310

What is a Database Management System?

- A software package that manages a database
  - Supports a high-level access language (e.g. SQL)
  - Supports concurrent access to very large amounts of data
- Also known as DBMS

Database System

- The DBMS software together with the data itself. Sometimes, the applications are also included



DBMS is a collection of related data.

True

False

Why don't we “program up” databases when we need them?

- For simple and small databases this is often the best solution
- We run into problems when
  - The structure is complicated (more than a simple table)
  - The database gets large
  - Many people want to use it simultaneously

Example: Personal Calendar

What	Day	When	Who	Where
Lunch	10/24	1pm	Rick	Joe's Diner
CS123	10/25	9am	Dr. Egghead	Morris234
Biking	10/26	9am	Jane	Jane's house
Dinner	10/26	6PM	Jane	Café Le Boeuf

Problem 1: Data Organization

What	When	Who-name	Who-email	Who-tel	….	Where
…						

“Link” Calendar with Address Book?

- Two conceptual “entities”: contact information and calendar with a relationship between them
- This link could be based on something as simple as the person's name

Problem 2: Efficiency

- Size of personal address book is probably less than one hundred entries, but there are things we'd like to do quickly and efficiently
  - “Give me all appointments on 10/28”
  - “When am I supposed to meet Jim?”
- “Program” these as quickly as possible and have these programs executed efficiently
- What would happen if you were using a “corporate” calendar with hundreds of thousands of entries?

Problem 3: Concurrency and Reliability

- Suppose other people are allowed access to your calendar and are allowed to modify it? How do we stop two people changing the file at the same time and leaving it in a physical (or logical) mess?
- Suppose the system crashes while we are changing the calendar. How do we recover our work?

Data Model

- A set of concepts to describe the structure of a database
  - Structure: data types, relationships, constraints, and operations
- **Conceptual** data models
  - High level
  - Provide concepts that are close to the way users perceive data
- **Physical** data models
  - Low level
  - Provide concepts that describe details of how data is stored in the computer
- **Implementation** data models
  - Provide concepts that fall between the above two, balancing user views with some computer storage details

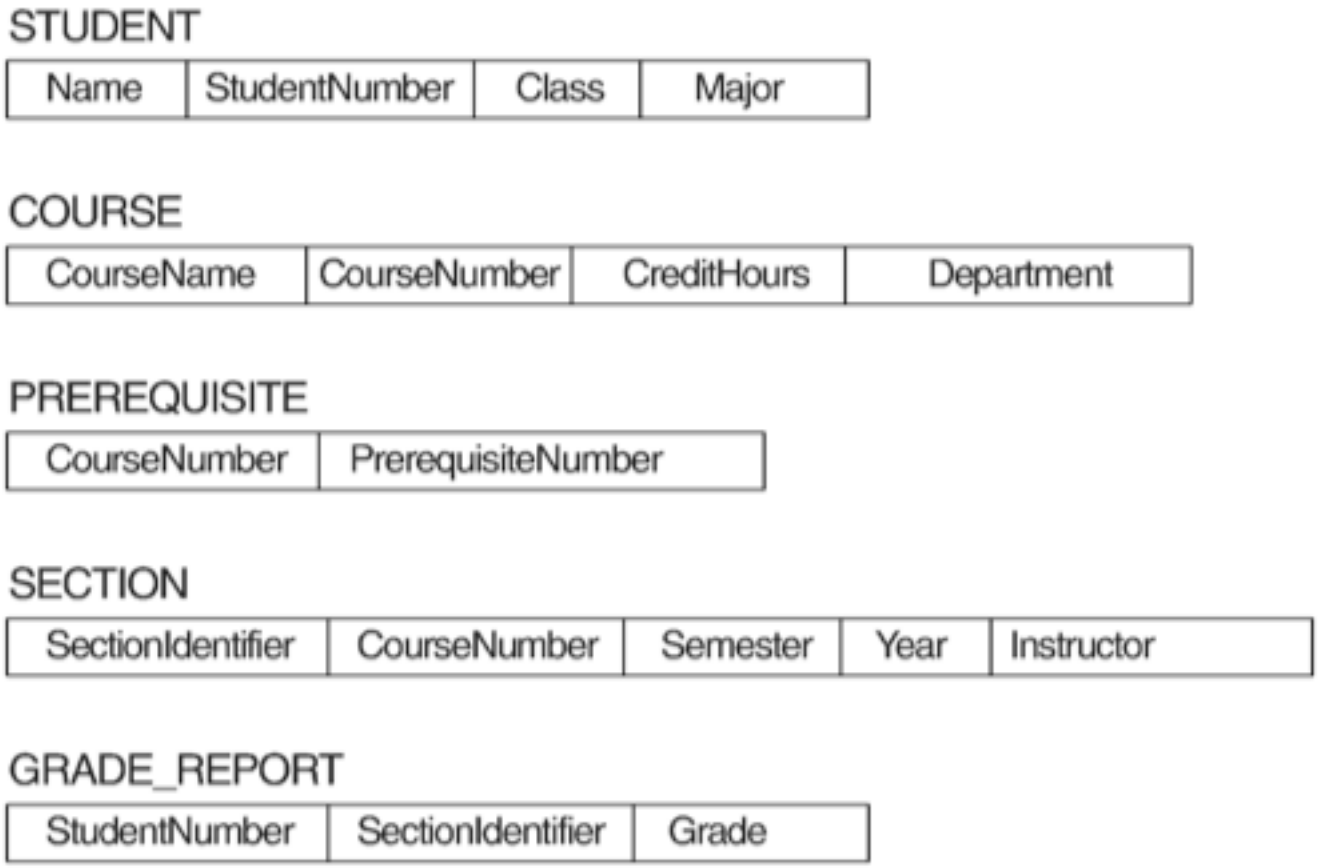
Database Schema

- The **description** of a database
- Includes descriptions of the database structure and the constraints that should hold on the database
- Specified during DB design
- Not expected to change frequently

Is database schema the same as database itself?

Database Schema Example

- Schema diagram for the student records and grades database



Instance, State

- **Database Instance**:
  - The actual data stored in a database at a particular moment in time
  - Also called **database state** (or **occurrence**)
- Schema vs. Instance (State):
  - The **database schema** changes very infrequently
  - Schema is also called **intension**
  - The **database state** changes every time the database is updated
  - State is called **extension**

The database schema changes whenever the database instance changes.

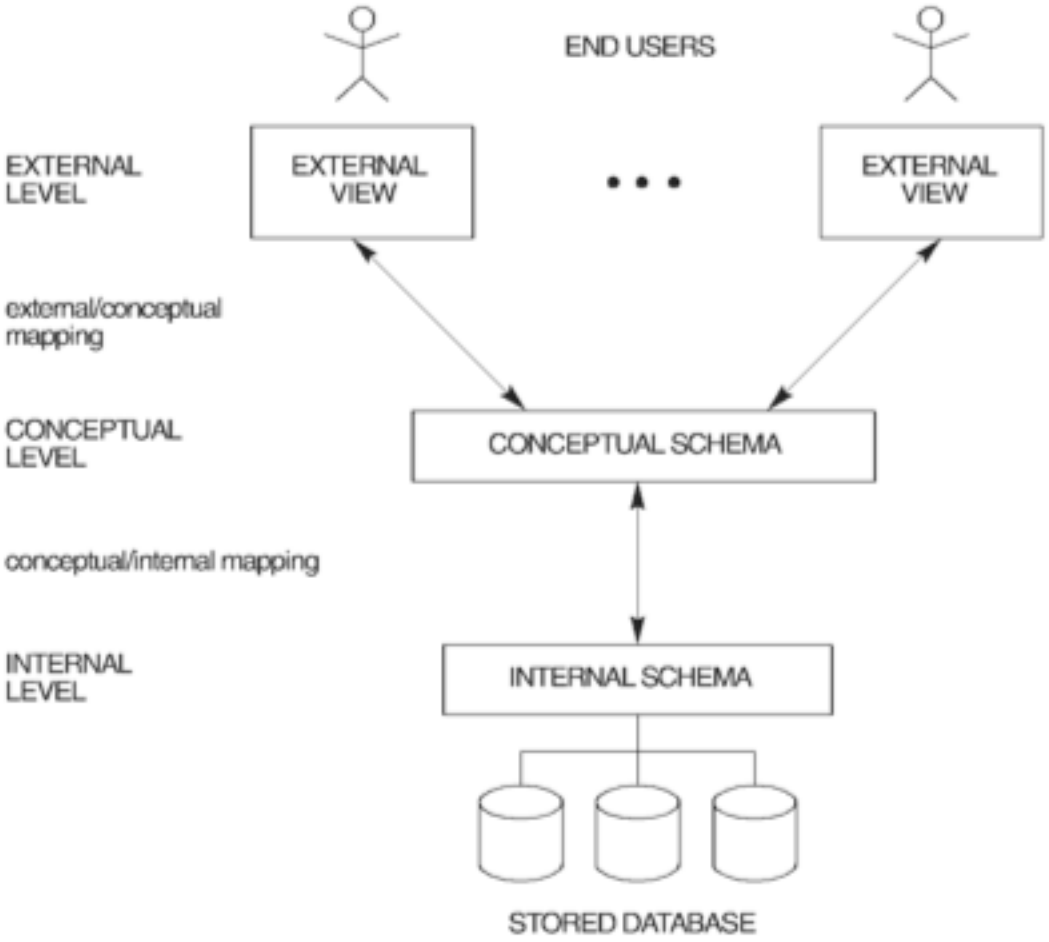
True

False

Three-Schema Architecture

- **Internal** schema
  - Describes data storage structures and access paths
  - Typically uses a physical data model
- **Conceptual** schema
  - Describes the structure and constraints for the whole database
  - Uses a conceptual or an implementation data model
- **External** schema
  - Describes the various user views
  - Usually uses the same data model as the conceptual level

The Three Schema Architecture



Data Independence

- Changing schema at one level of a database without having to change the schema at the next higher level
- A user of a relational database system should be able to use SQL to query the database without knowing about how precisely data is stored, e.g.

```
SELECT When, Where
FROM Calendar
WHERE Who = “Bill”;
```

More on Data Independence

- **Logical data independence**
  - Capacity to change conceptual schema without having to change external schema or application programs
  - Protects the user from changes in the logical structure of the data
- **Physical data independence**
  - Capacity to change the internal schema without having to change the conceptual (external) schemas
  - Protects the user from changes in the physical structure of data

DBMS Languages

- DDL: specifies database schema
- DML: enables users to access or manipulate data (retrieve, insert, replace, delete)
  - Procedural
    - Describes what data is needed and how to get it
    - Relational algebra
    - Low-level
  - Non-procedural
    - Describes what data is needed without specifying how to get it
    - SQL
    - High-level

Advantages of using a DBMS

- Representing complex relationships among data
- Efficient data access
- Supports concurrent access and crash recovery
- Data abstraction
- Data independence
- Enforcing integrity and security

When not to use a DBMS

- Main costs of using a DBMS:
  - High initial investment and possible need for additional hardware
  - Overhead for providing security, recovery, integrity, and concurrency control
- When a DBMS may be unnecessary:
  - Simple, well defined, and not expected to change
  - If access to data by multiple users is not required

The DBMS Marketplace

- Relational DBMS: Oracle, MySQL, IBM DB2, Microsoft SQL Server, Microsoft Access,…
- Relational companies were challenged by “object-oriented DB” companies in the 90s
- But countered with “object-relational” systems, which retain the relational core while allowing type extension as in OO systems
- Relational companies are also challenged by NoSQL companies

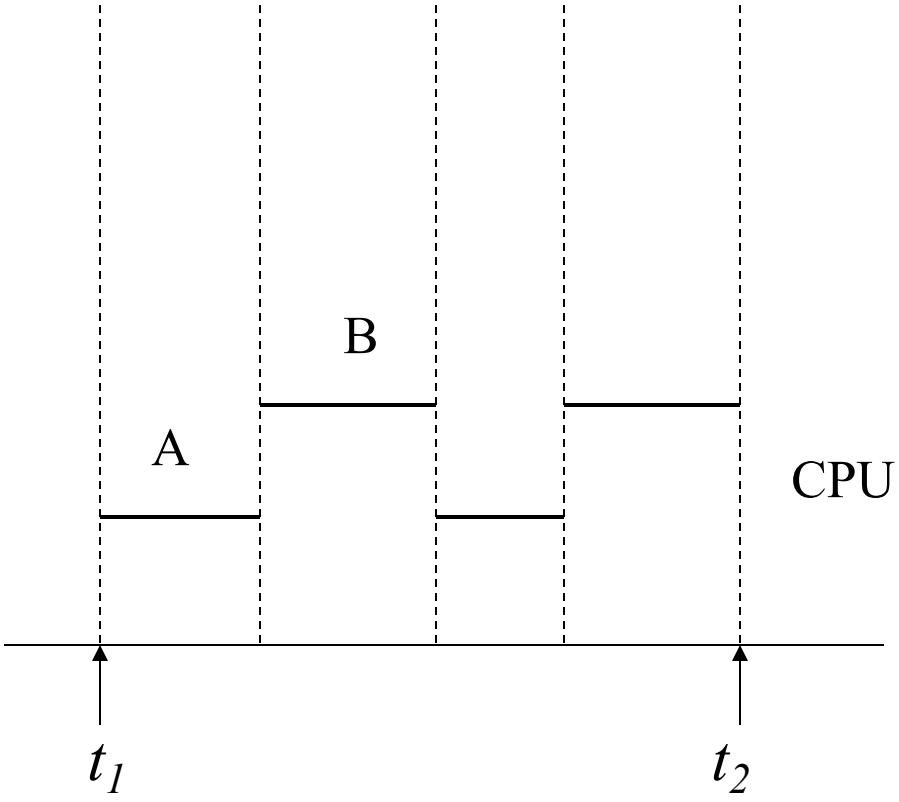
People Associated with a Database

- Database administrator (DBA)
  - Defines schemas, enforces security and authorization, maintains data availability, and recovers from failures
- Application programmers
  - Write programs and make it available to the end-users
- Sophisticated users
  - Use SQL to access the database interactively
- Naive users
  - Invoke application programs

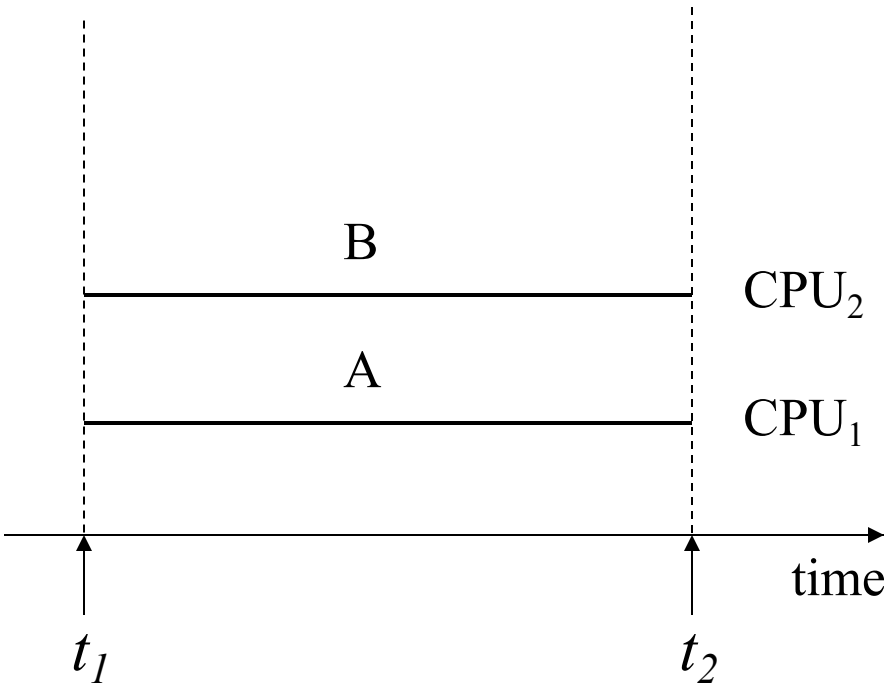
Why Do We Need Transactions?

- It's all about fast query response time and correctness
- DBMS is a multi-user systems
  - Many different requests
  - Some against same data items
- Figure out how to interleave requests to shorten response time while guaranteeing correct result
  - How does DBMS know which actions belong together?
- Solution: Group database operations that must be performed together into transactions
  - Either execute all operations or none

Concurrent Transactions



interleaved processing



parallel processing

Terminology

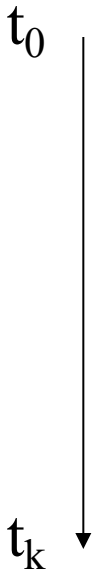
- A transaction T is a logical unit of database processing that includes one or more database access operations
- Basic database access operations: read_item (X) and write_item (X)
  - Embedded within application program
  - Specified interactively (e.g., via SQL)

Sample Transaction (Informal)

- Example: Move $40 from checking to savings account
- To user, appears as one activity
- To database:
  - Read balance of checking account: read(X)
  - Read balance of savings account: read (Y)
  - Subtract $40 from X
  - Add $40 to Y
  - Write new value of X back to disk
  - Write new value of Y back to disk

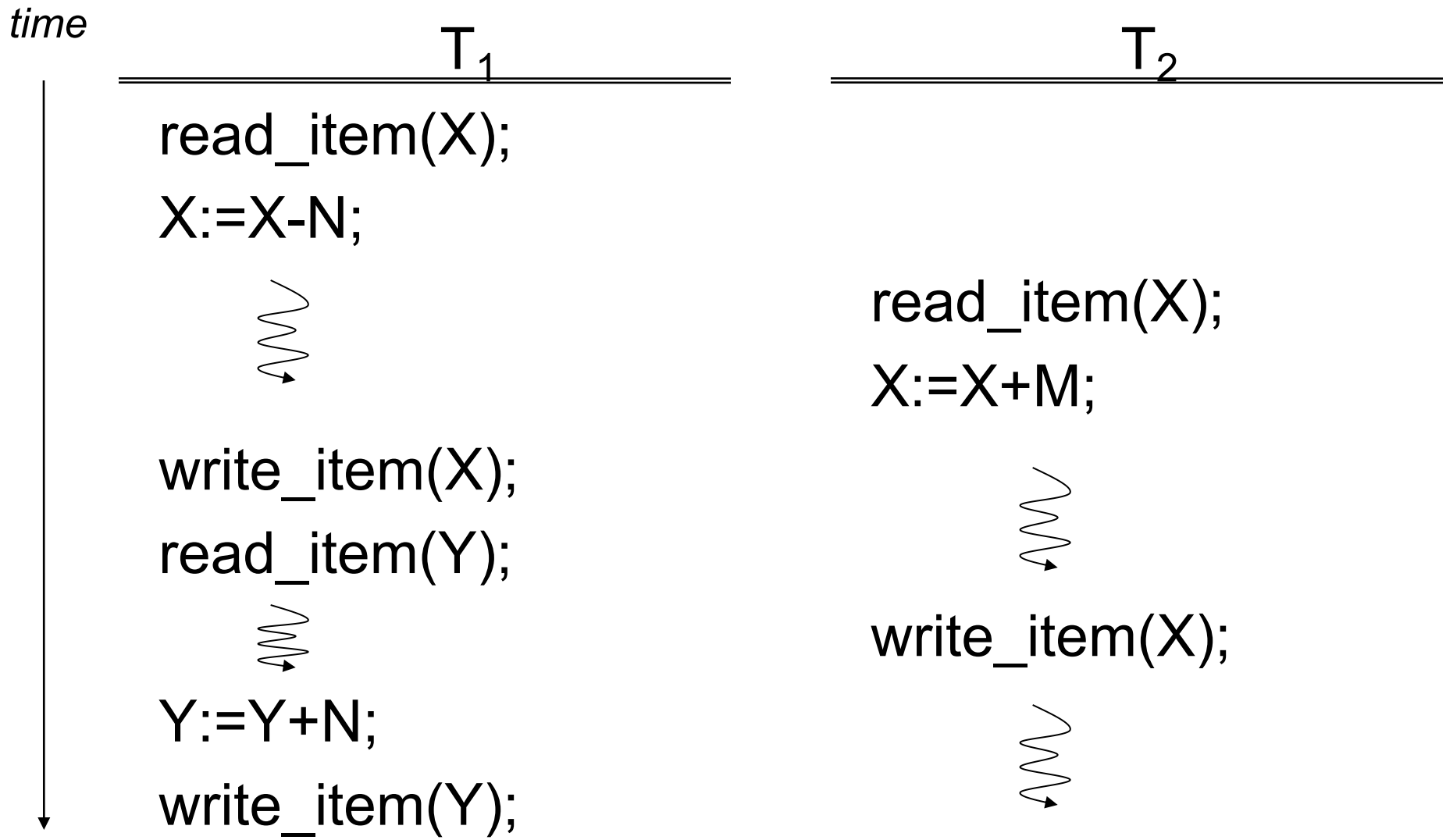
Sample Transaction (Formal)

T_1

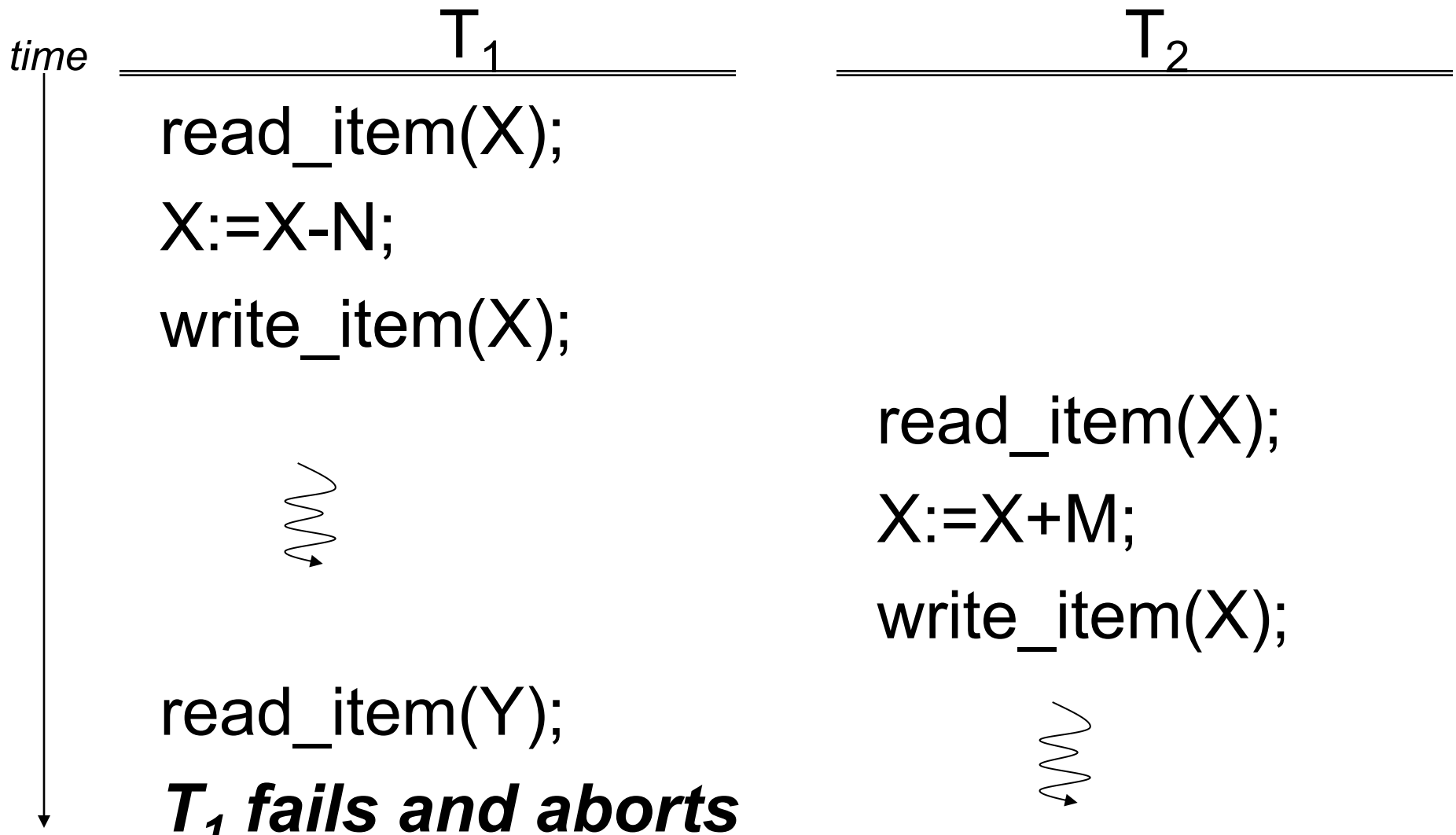



```
read_item(X);
read_item(Y);
X:=X-40;
Y:=Y+40;
write_item(X);
write_item(Y);
```

Lost Update Problem



Temporary Update (Dirty Read)



ACID Properties of Transactions

- *Atomicity:* Transaction is either performed in its entirety or not performed at all
  - Task of the recovery subsystem to enforce atomicity
- *Consistency preservation:* Transaction must take the database from one consistent state to another
  - Users/DBMS: enforce implicit and explicit constraints
- *Isolation:* Transaction should appear as though it is being executed in isolation from other transaction
  - Enforced by concurrency control subsystem
- *Durability:* Changes applied to the database by a committed transaction must persist
  - Enforced by recovery subsystem

NoSQL

- Four categories:
  - Key-value database
  - Column-family database
  - Document database
  - Graph database
- Data model
  - Flexible data model: aggregates and etc.
  - Support big data
- Schemaless database
  - Change data storage easily
  - Nonuniform data
  - Schemas have value

Why is RDBMS still relevant?

- Ease of use: the use of tables to store data in columns and rows make it easy to access and manage data
- Data security: you can hide sensitive tables and give them their authorization codes, providing a layer of protection for your data
- SQL standard: SQL is a standardized language well understood by many applications, and many of the alternative database options provide SQL interface
- Data integrity: the structure of the relational database preserves the integrity of the data and makes it easier to meet compliance regulations
- Performance: uses indexes to sort data and speed up performance, and supports both desktop and web applications
- Development and support: the large players - Oracle, Microsoft - have a vested interest in continuing to develop and evolve their database offering to meet modern standards
- RDBMS standards: relational databases adhere to ACID properties to ensure the reliability of transactions

Which of the following is not an advantage of RDBMS?

Support concurrent access

Data independence

Schemaless