

Lecture 1: Introduction

CS348 Spring 2025:
Introduction to Database Management

Instructor: **Xiao Hu**
Sections: 001, 002, 003

About the Instructor

- Instructor: **Xiao Hu**
 - Assistant Professor in the School of Computer Science
 - Email: xiaohu@uwaterloo.ca
 - <https://cs.uwaterloo.ca/~xiaohu/>
 - Office hour: Tuesdays, 10 AM- 11 AM
- Why CS348: intro to database management?
- What is my expectation?
- What is your expectation?

Outline For Today

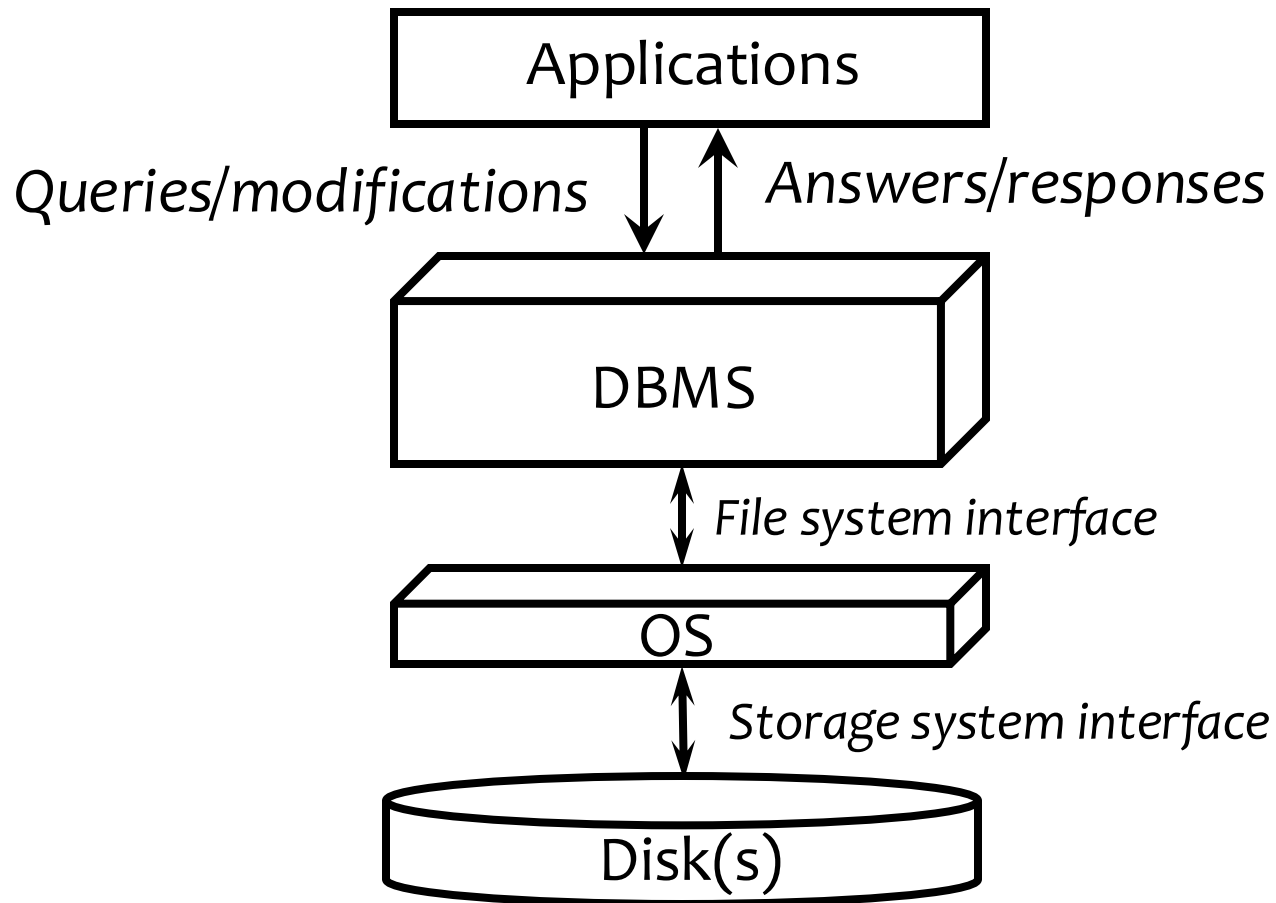
- Overview of database management systems (DBMS)
 - Challenges with data management
 - How DBMS help overcome these challenges
- Administrative Information

So, what is a DBMS?

From Oxford Dictionary:

- **Database**: an organized body of related information
- **Database system, DataBase Management System (DBMS)**: a software system that facilitates the creation, maintenance, and use of an electronic database

What is a DBMS?

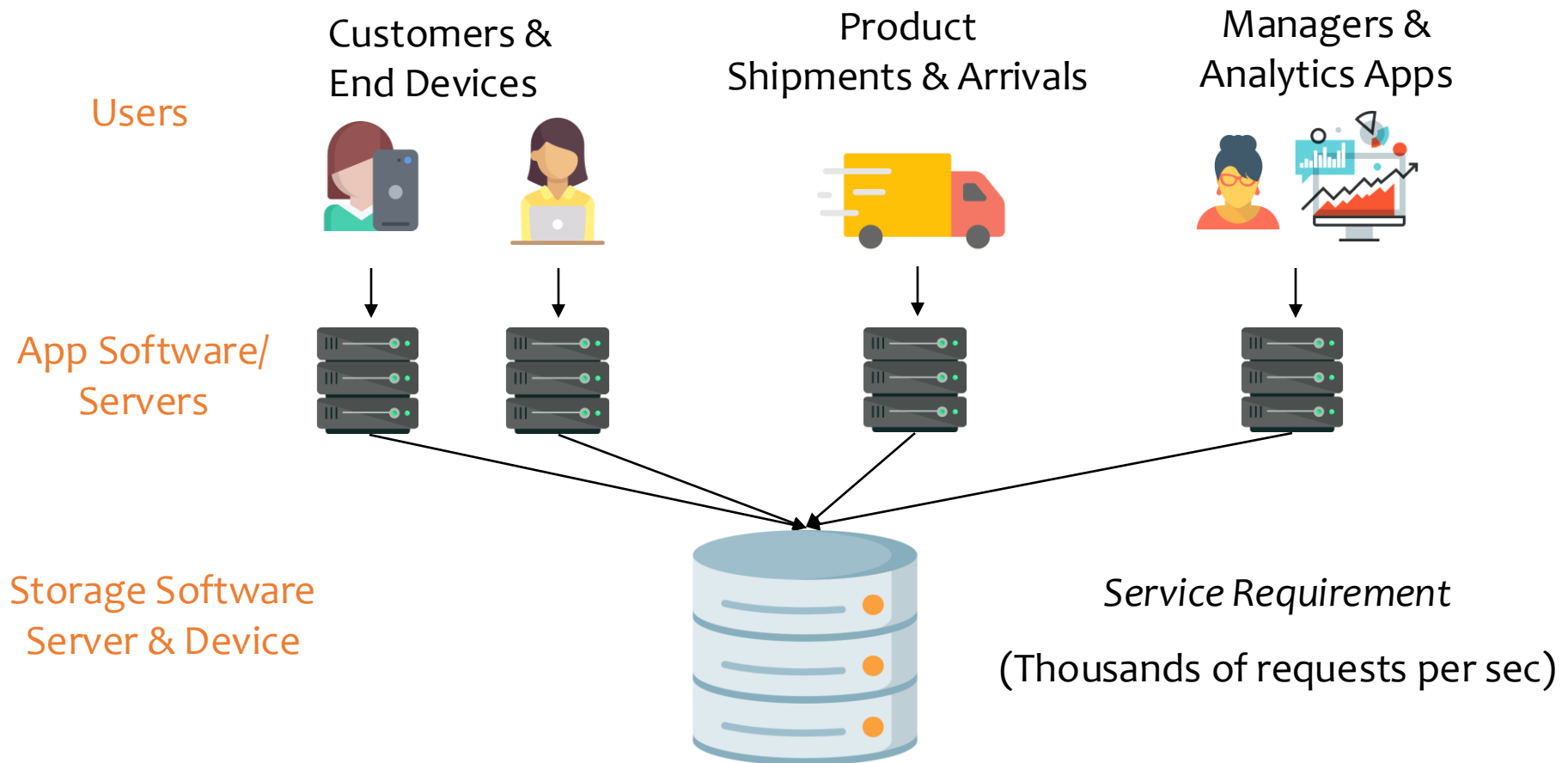


Why Do App Developers Need a DBMS?

- Challenges to overcome if we do NOT use a DBMS:
 - Physical data design
 - Query processing
 - Data integrity
 - Concurrency
 - Recovery and Backup

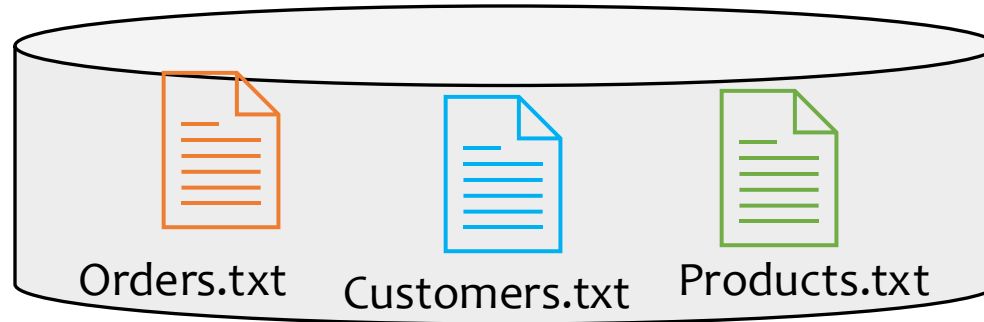
Why Do App Developers Need a DBMS?

- Application: Order and Inventory Management in E-commerce



Write Storage Software in Java/C++?

- Directly use the file system of the OS.
 - One or more files for orders, customers, products etc.



- Possible problems in **physical data design**
 - For each customer store **name**, **birthdate**
 - How many bytes for each record?
 - Encoding of string names? Fixed or variable length?
 - How to quickly find a record?

Physical Data Design

- Variable-length design

name-len (bytes)	name payload	birthdate (fixed 4 bytes)
------------------	--------------	---------------------------

11	Alice Smith	2001/09/08	19	Alexander Desdemona	2002/05/20
6	Ali Jo	1992/02/25	26	Montgomery Cambridgeshire	1992/02/25
...

- Fixed-length design

Overflow ptr	len	name (16 byte)	birthdate (4 bytes)
--------------	-----	----------------	---------------------

null	11	Alice Smith -----	2001/09/08	ona	idgeshire
0	19	Alexander Desdem	2002/05/20
null	6	Ali Jo -----	1992/02/25			
1	26	Montgomery Cambr	1992/02/25			

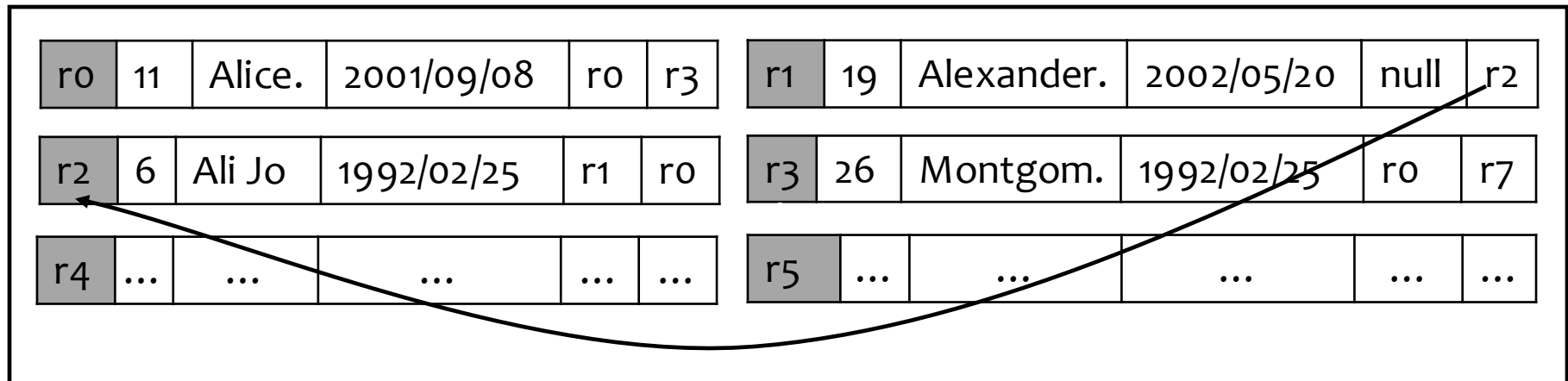
Customers.txt

Customer-overflow.txt

Physical Data Design

- Chained Design: Maybe to keep in sorted alphabetical order

name-leng (bytes)	name payload	birthdate (fixes 4 bytes)	prev ptr	next ptr
-------------------	--------------	---------------------------	----------	----------



Customers.txt

Many design options and difficult for app developers!

Physical data design should be independent!

Query Processing

- Who are the top-paying customers?
- Compute total sales by customer (assume fixed-length records)

```
file = open("Orders.txt")
HashTable ht;
for each line in file:
// some code to parse custID and price
if (ht.contains(custID))
    ht.put(custID, ht.get(custID) + price)
else: ht.put(custID, price);
file.close();
```

O1	Cust1	BookA	\$20
O2	Cust2	WatchA	\$120
O3	Cust1	DiapersB	\$30
O4	Cust3	MasksA	\$15
...
...

Orders.txt

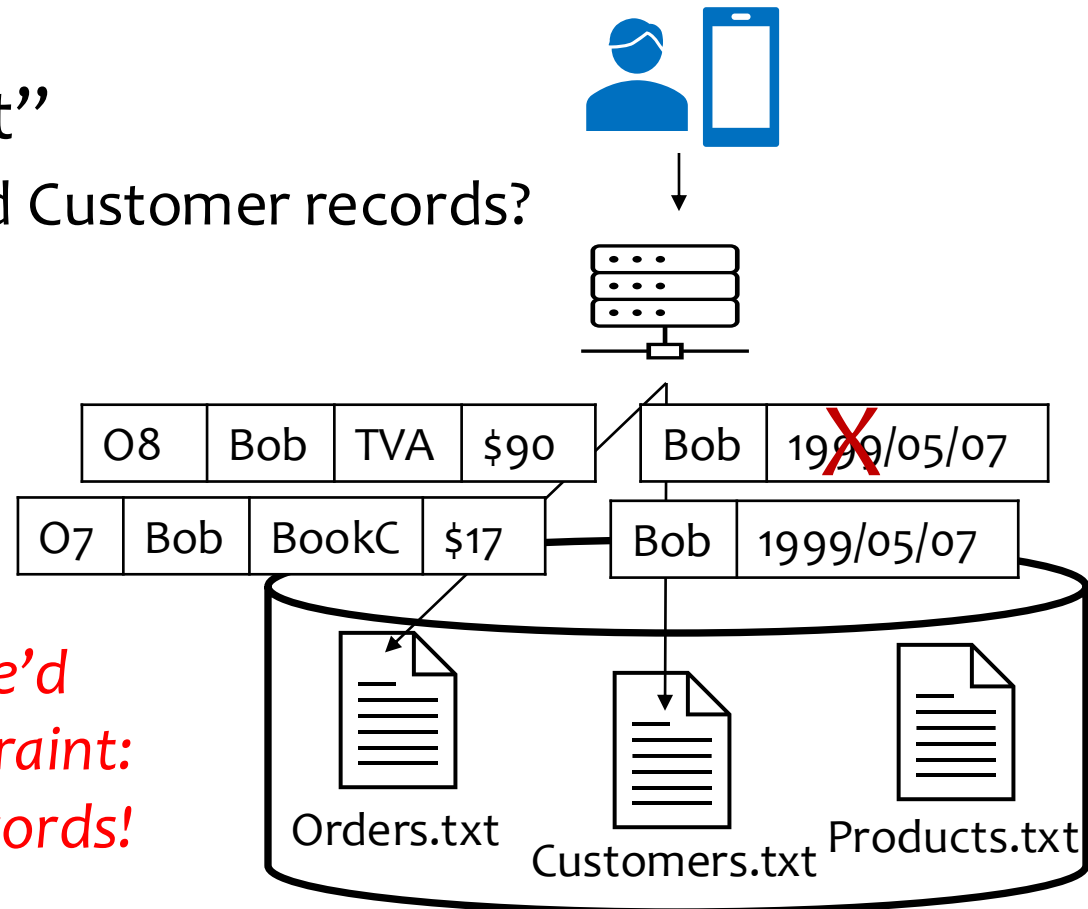
How to reuse it for other queries, for other physical data layouts?
Writing an algorithm for each task won't scale!

Query Processing

- Many, many more important business analytical questions:
 - List of Orders that bought a product that cost $> \$500$
 - Last Order from Cust4?
 - Who are the closest co-purchasers of Cust4?
- There are numerous possible algorithms and implementations; it is difficult to choose
 - Sorting, hashing, or building indexes
 - What is the cost of each possible algorithm?
 - How to choose the “optimal” one?
- Query execution details should be independent!

Integrity

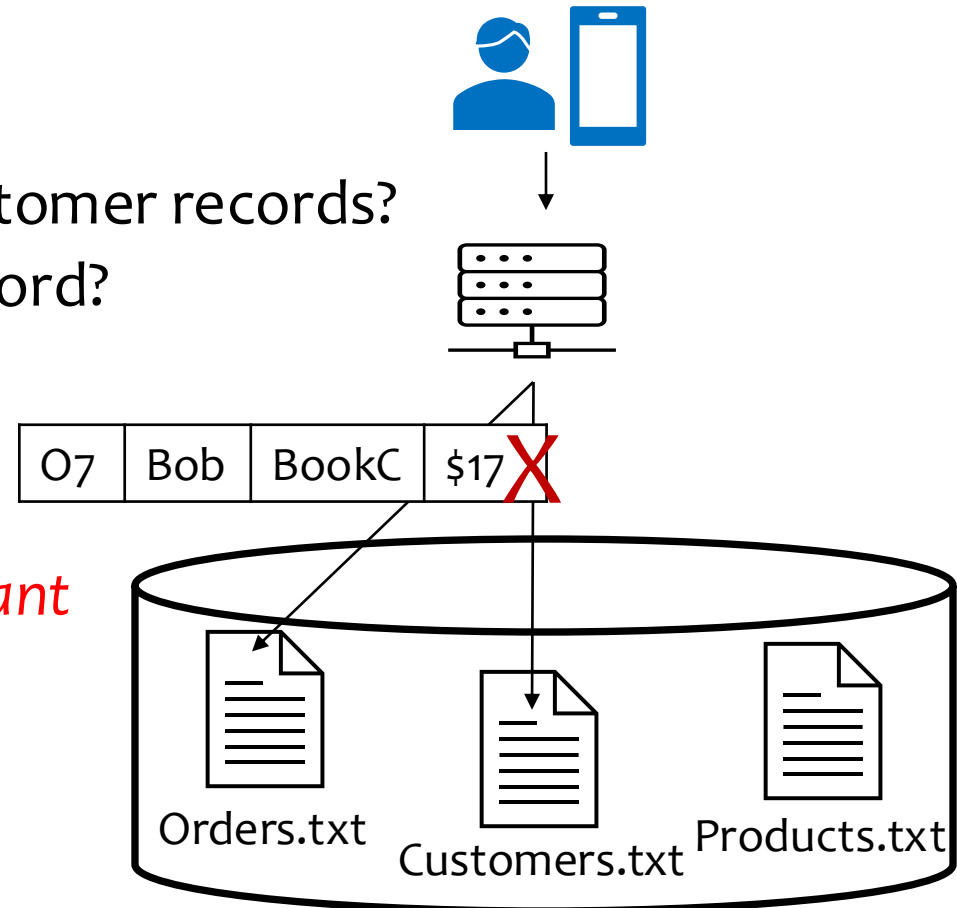
- Many ways data can be corrupted
 - Often: Wrong application logic or bugs in application
- “Checkout As Guest”
 - Writes the Order and Customer records?



Likely an inconsistency. We'd want to enforce the constraint: No duplicate customer records!

Integrity

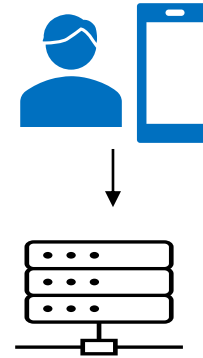
- Many ways data can be corrupted
 - Often: Wrong application logic or bugs in application
- “Checkout As Guest”
 - Writes the Order and Customer records?
 - Only writes the Order record?



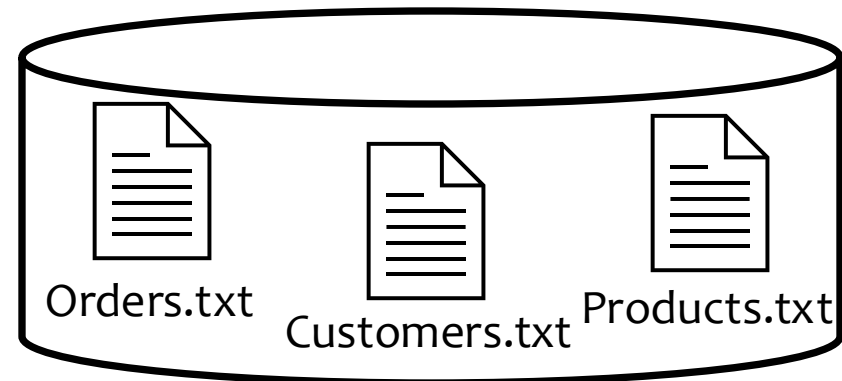
Likely an inconsistency. We'd want to enforce the constraint: Every order's customer record exists!

Integrity

- Many ways data can be corrupted:
 - Often: Wrong application logic or bugs in application
- “Checkout As Guest”
 - Writes the Order and Customer records?
 - Only writes the Order record?



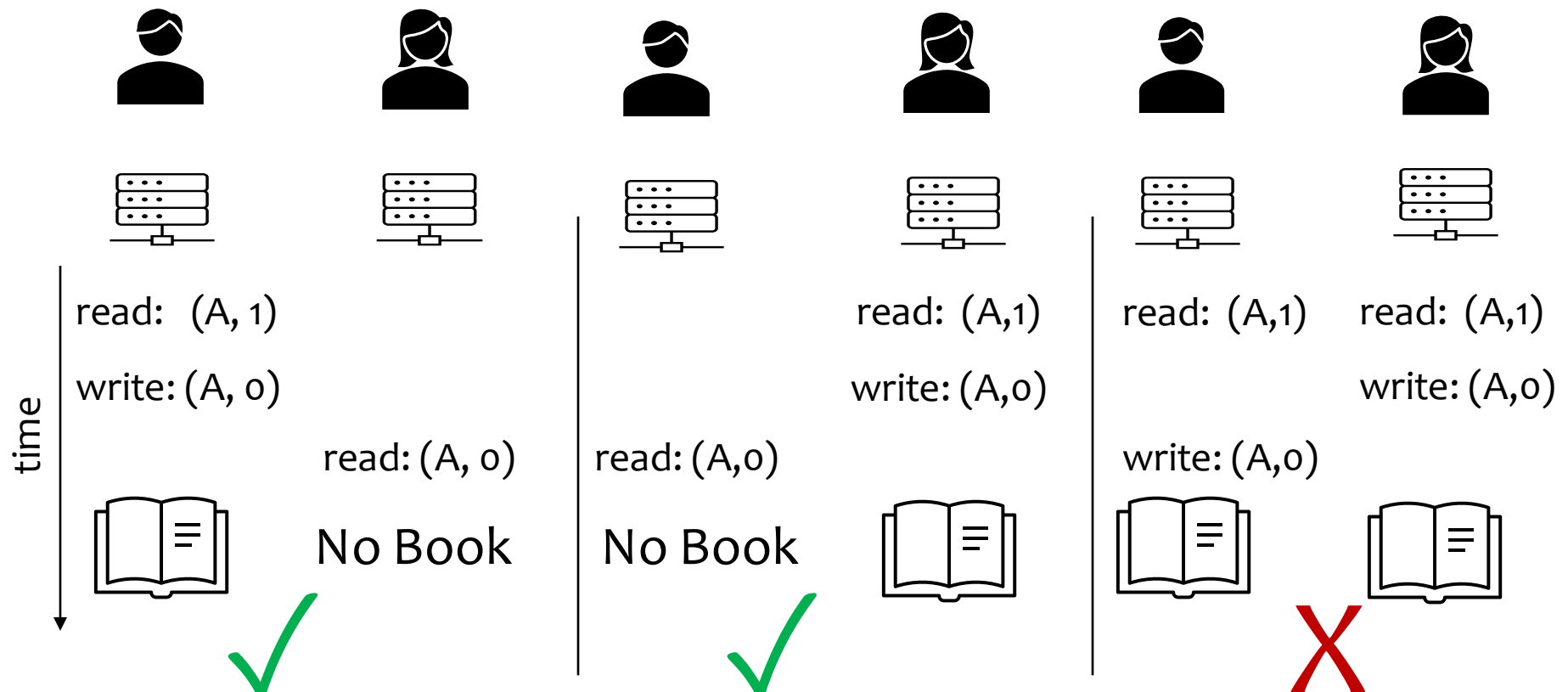
Incorrectly handling consistency
violates data integrity!



Concurrency

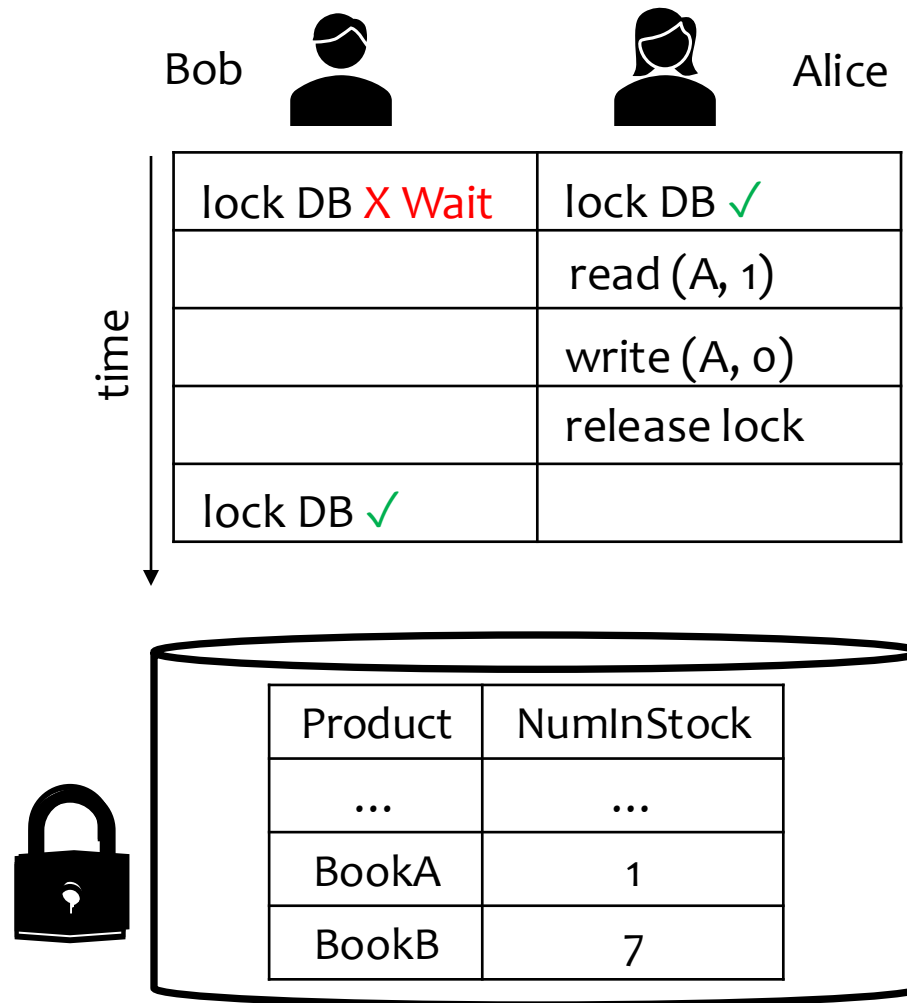
```
Buy_Product_Subroutine(string prodName):
(prod, numInStock) = readProduct(prodName)
if (numInStock > 0):
    writeProduct (prod, numInStock - 1)
else throw ("Cannot buy product!");
```

- Alice and Bob concurrently order BookA
 - suppose 1 left in stock



Concurrency: Global DB Lock

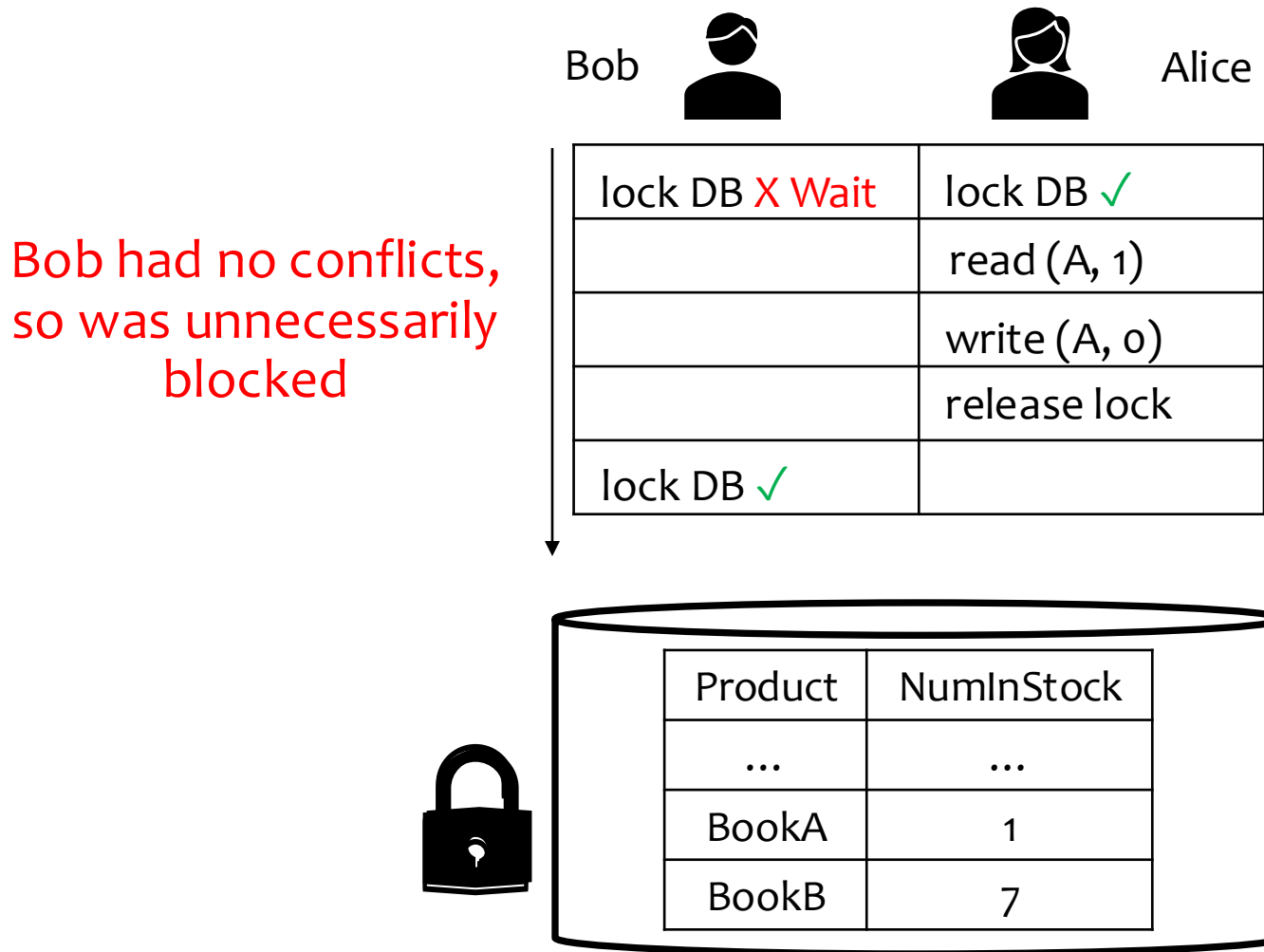
- Both Alice and Bob order BookA



Safe but inefficient!

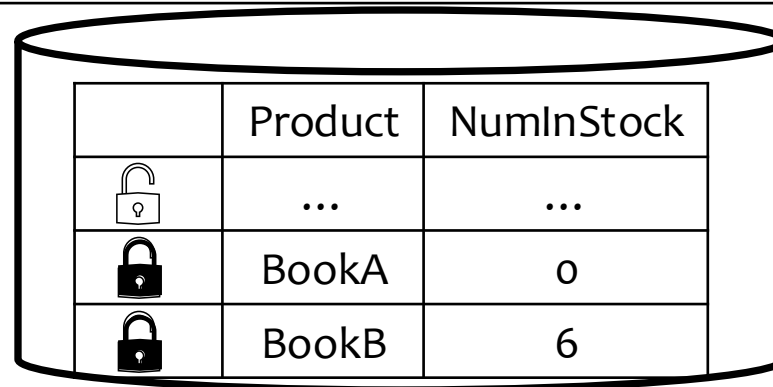
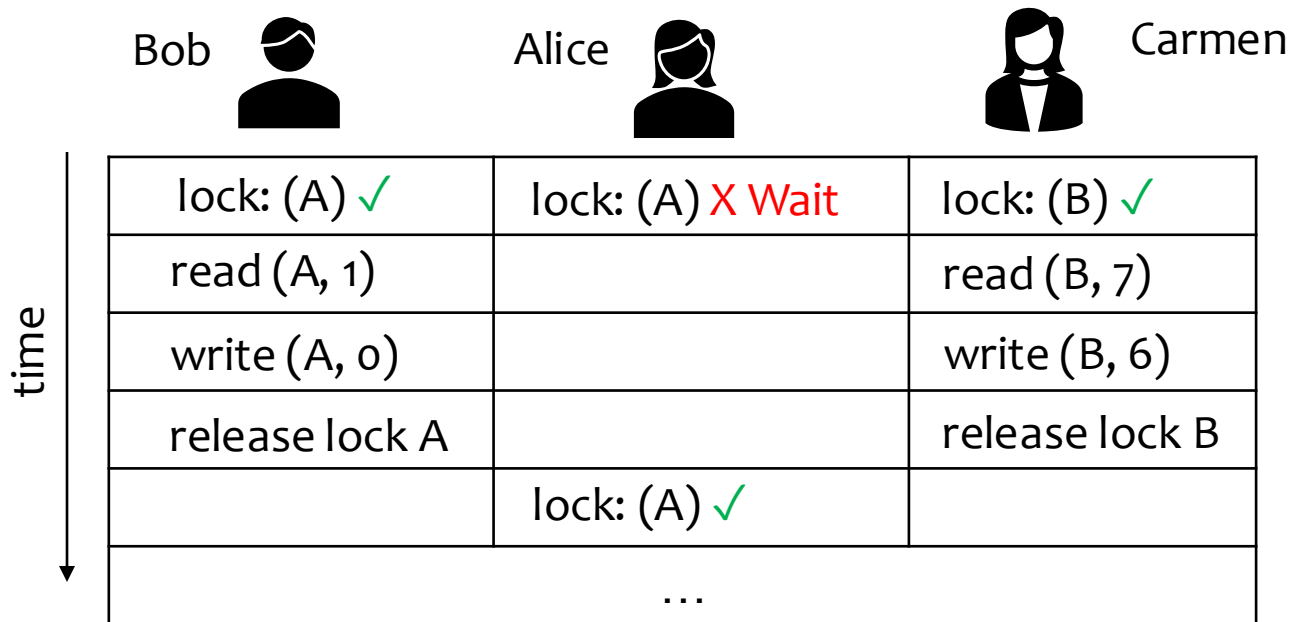
Concurrency: Global DB Lock

- Alice orders BookA and Bob orders BookB



Concurrency: Record-level Lock

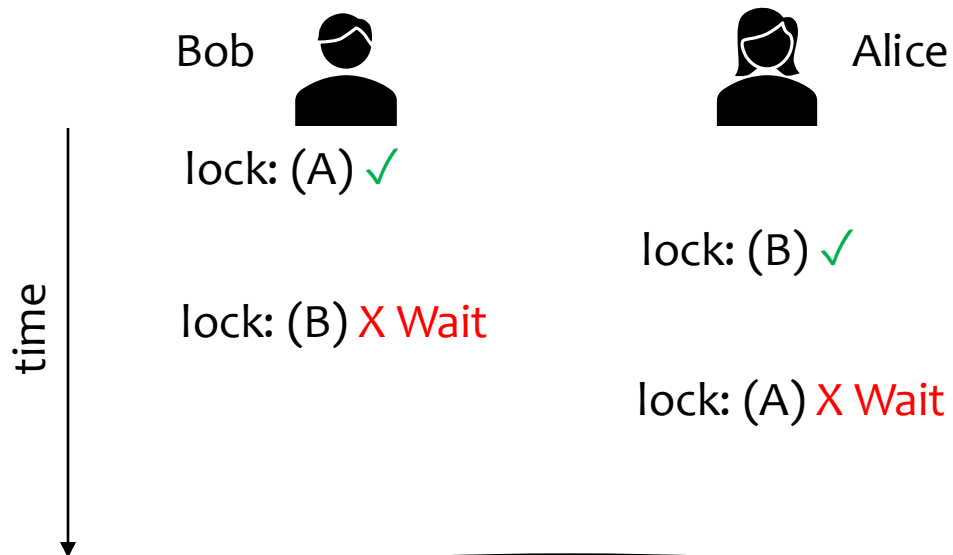
- Alice, Bob order BookA and Carmen orders BookB






Safe and Efficient.
What can go wrong?

Concurrency: Deadlocks!

- Alice, Bob order BookA and BookB together



	Product	NumInStock

	BookA	1
	BookB	7


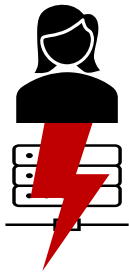
How can we detect and avoid deadlocks?

Concurrency

- What is a correct/incorrect state upon concurrent updates to data?
- What protocols or algorithms can ensure a correct state?
- How to guarantee correctness while ensuring efficiency?

Recovery and backup

- What if your disk fails in the middle of an order?
- What if your server software fails due to a bug?
- What if a power outage in the machine storing files?

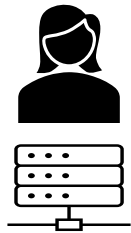


Product	NumInStock
...	...
BookA	1
BookB	7

Recovery and backup

- What if your disk fails in the middle of an order?
- What if your server software fails due to a bug?
- What if a power outage in the machine storing files?

Suppose Alice orders both BookA and BookB



write (A, 0)

write (B, 6)

*Failure happens before (B, 6) is written!
Inconsistent state!
How to recover from inconsistent state?*

A diagram of a database cylinder. On the left side, there is a red lightning bolt. On the right side, there is a large red 'X'. Inside the cylinder is a table.

Product	NumInStock
...	...
BookA	0
BookB	7

Product	NumInStock
...	...
BookA	0
BookB	6

Summary of challenges

- Physical data design
- Query processing
- Data integrity
- Concurrency
- Recovery and backup

A database management system (DBMS) helps us solve all the discussed problems

The birth of DBMS

- History of Database:

<https://www.youtube.com/watch?v=KG-mqHoXOXY>

(from Computer History Museum)

- We will focus on Relational DBMS (RDBMS)

Data Model

- Relational Model
 - Data is modeled as a set of relations or tables
 - Much higher-level abstraction than bits/bytes

Customer

name	birthdate
Alice	2001/09/08
Bob	1999/05/19
...	...

Orders

orderID	cust	product	price
001	Alice	Book	20
002	Bob	Beer	15
...

Product

product	numInStock
Book	20
Beer	15
...	...

```
CREATE TABLE Customers
  name varchar(255),
  birthdate DATE;
```

The RDBMS takes care of physical record design: Fixed-length/var-length, columnar, row, chained etc.

Data Model

- Physical Data Independence:
 - Throughout the lifetime of the application, the RDBMS can change the physical layout for performance or other reasons and the applications is oblivious to this and continues working as-is.
 - Example: A new column can be added that changes the record design
 - Example: A compressed column can be uncompressed

Relational data model delegates the responsibility of physical data design and access to these records to the DBMS

High-level Query Language

- SQL: Structured Query Language
- **Declarative:** you can describe the output of the computation but not how to perform the computation
- Recall managers' question: Who are top paying customers?

Orders

orderID	cust	product	price
001	Alice	BookA	20
002	Bob	BookB	15
...

```
SELECT cust, sum(price) as sumPay  
FROM Orders  
ORDER BY sumPay DESC
```

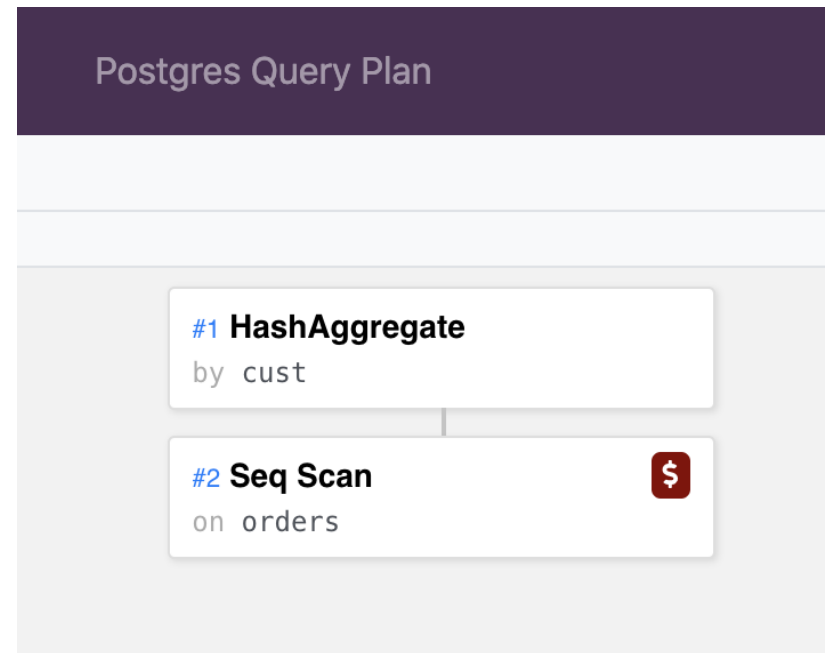
- No procedural description of how to execute the query

Query Optimizer

- DBMS automatically generates an efficient algorithm for executing the query:

```
SELECT cust, sum(price) as sumPay  
FROM Orders  
ORDER BY sumPay DESC
```

Query optimizer delegates the responsibility of finding an efficient algorithm to execute the query



Integrity

- Recall the bug in “Checkout As Guest”
- In RDBMS: add uniqueness constraints (primary key constraints)
 - Writes the Customer record
 - Assume Bob shops again
 - (Bob, 1999/05/07) is duplicated!

```
CREATE TABLE Customers  
(name varchar(255),  
DOB DATE,  
PRIMARY KEY (name));
```

```
template1=# INSERT INTO Customers Values ('Bob', '1999/05/07');  
INSERT 0 1  
template1=# INSERT INTO Customers Values ('Bob', '1999/05/07');  
ERROR:  duplicate key value violates unique constraint "customers_pkey"  
DETAIL:  Key (name)=(Bob) already exists.
```

DBMS enforces the constraint and maintain the data integrity at all times on behalf of the applications

Concurrency

DBMS will use different transaction protocols to ensure safe concurrency of queries

(Simplified) SQL:

BEGIN TRANSACTION

UPDATE Products

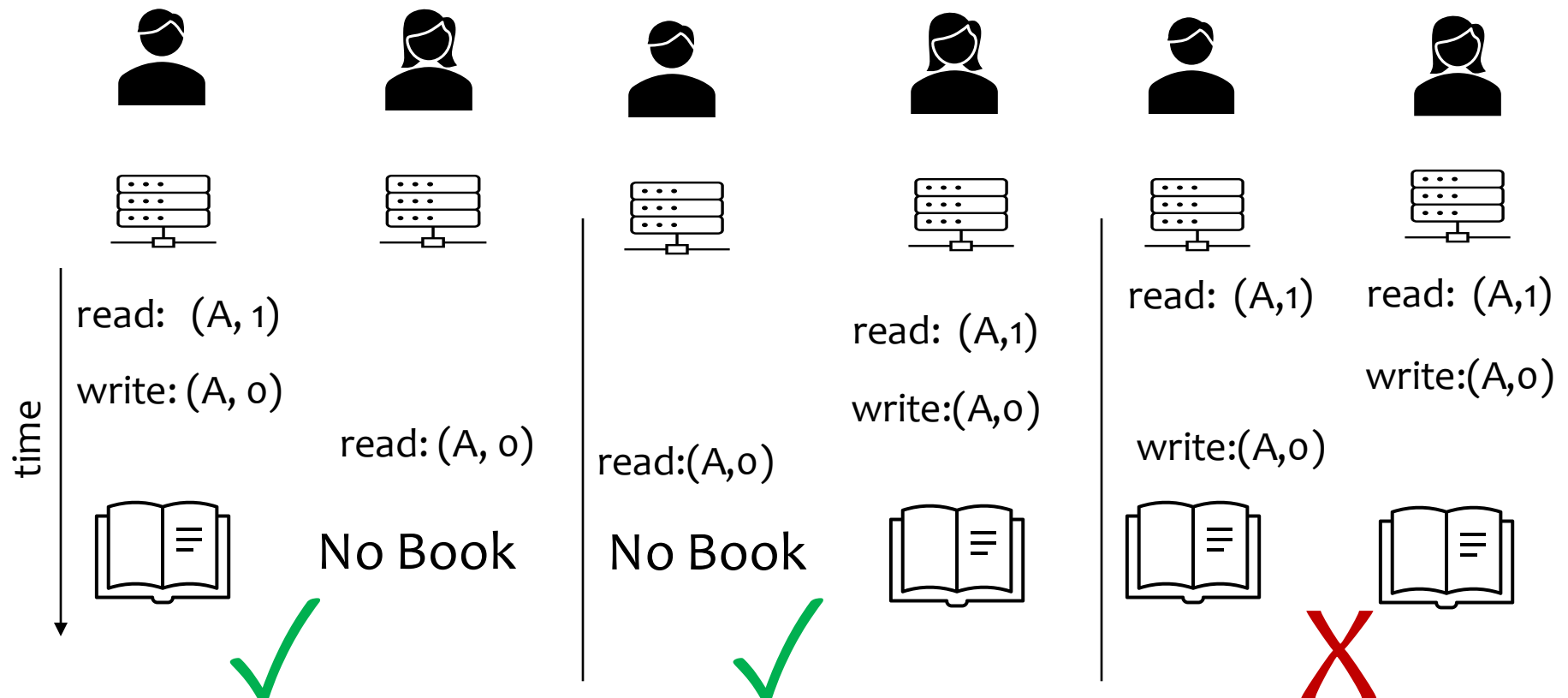
SET numInStock = numInStock - 1

WHERE name = "BookA"

INSERT INTO Orders

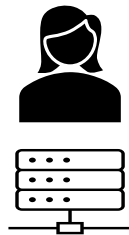
VALUES ("Alice", "BookA", \$20)

COMMIT




Backup and Recovery


- Failure scenario: Alice orders both BookA and BookB
- Suppose a power failure occurs and the DBMS fails in the middle of committing the transaction



write (A, 0)
write (B, 6)



Product	NumInStock
...	...
BookA	0
BookB	7



DBMS uses checkpointing
and logging to undo partial
changes and revert back to
a consistent state

Product	NumInStock
...	...
BookA	1
BookB	7



Summary

DBMS is an indispensable core system software to develop any application that stores, queries, and processes data.

A Glimpse of Current DBMS Market



Hundreds of companies producing DBMSs: Many RDBMS/SQL, but also graph, RDF, Document DB, Key-value stores etc..

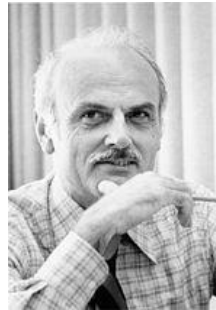
4 Turing Award Winners!

- Charles Bachman, 1973



Introduced DB Systems

- Edgar F. Codd, 1981



High-level/Declarative Programming:
Relational Data Model & Algebra

- Jim Gray, 1998



Transactions:
concurrent data-manipulation

- Michael Stonebraker, 2014



Relational DBMS
(e.g. Ingres, Postgres) and
modern DBMSs
(e.g. C-store, H-store, SciDB)

Outline For Today

- Overview of database management systems (DBMS):
 - Challenges with data management
 - How DBMSs help overcome these challenges
- Administrative Information

Course components

- **Relational databases** (Lectures 1-12)
 - Relational algebra
 - SQL
 - Database design
- **Database internals** (Lectures 13-20)
 - Storage
 - Indexing
 - Query processing and optimization
 - Transactions
 - Concurrency and Recovery

More about the Teaching Team

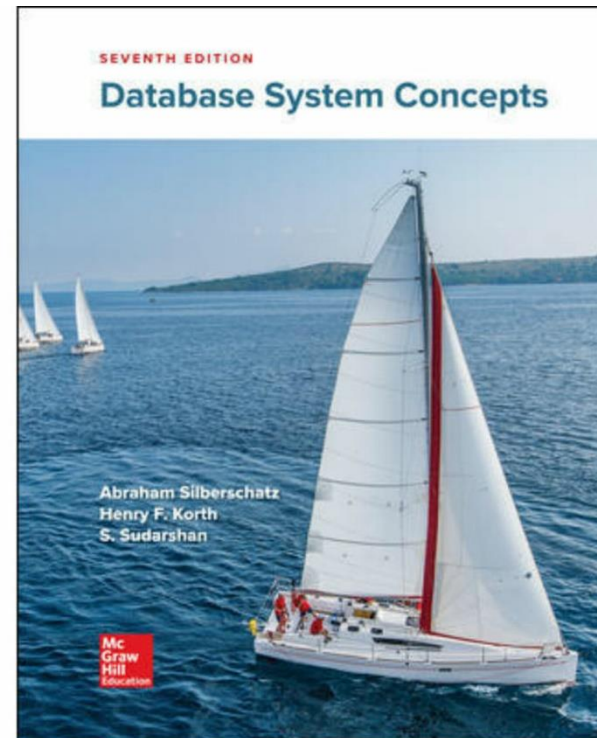
- Instructional support coordinator: **Sylvie Davies**
 - Email: sldavies@uwaterloo.ca
- IAs and TAs
 - Guy Coccimiglio (IA)
 - Nimmi Rashinika Weeraddana (IA)
 - Lasantha Fernando
 - Shufan Zhang
 - Mayank Jasoria
 - Zhengyuan Dong
 - Boyi Li
 - Amin Rezaei
 - Zhiang Wu
 - Christina Li
- Guest Lecture (Week 8): **Chao Zhang**
- Office Hours will be posted on Piazza/Learn

Who to reach out to?

- Lecture-related questions: reach out to me
- Assignment-related (such as regrade requests) and Project-related questions: reach out to IA and the respective TA
- Approved regrade, Late policy, Verification of illness: reach out to Sylvie and cc me
 - Must email Sylvie for accessibility/short term absence!
It's your responsibility to email Sylvie since I do not track them.

Textbook

- **Database System Concepts** (Seventh Edition)
Abraham Silberschatz, Henry F. Korth and
S. Sudarshan, McGraw Hill.
- A hard copy in Library



Logistics

- Course Website:
 - <https://student.cs.uwaterloo.ca/~cs348/>
 - Course schedule, lecture notes
- Learn:
 - <https://learn.uwaterloo.ca/>
 - Assignment questions, Partial solutions, Project info
- Piazza for student discussion, Q&A, TAs info:
 - <https://piazza.com/uwaterloo.ca/spring2025/cs348spring2025>
 - For student-student discussions (24-hour policy)
- Work submission: Crowdmark/Marmoset/Learn
 - Watch your emails for the links

Marking and Late Policies

- Marking and appeals:
 - For everything, there will be **an appeal deadline** that will be indicated on the front page
 - No appeals will be accepted past this date unless you were sick the entire period until the appeal date
- Late assignments/project deliverables
 - Late assignments will be accepted for **48 hours** past the due date, but...
 - For **each 24 hour** past the due date, a **5% penalty** will be applied (cumulatively) for assignments
 - For **each 24 hour** past the due date, a **25% penalty** will be applied (cumulatively) for projects

Assessments

- 3 Assignments
- 1 Midterm Exam (Jun 27, 4:30pm – 6pm)
- 1 Final Exam
- Group Project (Optional): Choose 1 mark breakdown
- But both exams are mandatory!

Mark Breakdown	Project-based	Exam-based
3 Assignments	30%	30%
Midterm Exam	15%	30%
Final Exam	20%	40%
Project	35%	-

Any use of GenAI in the assessments must be cited. You are accountable for the content and accuracy of all work you submit in this class.

Lecture

- Lecture slides released on Course Website
- Lecture format:
 - Important announcements (Don't miss this!)
 - Key points and Examples
 - Exercises with partial solutions
- Will be using some lecture materials from Prof. Jun Yang, Prof. Xi He and Prof. Sujaya Maiyya

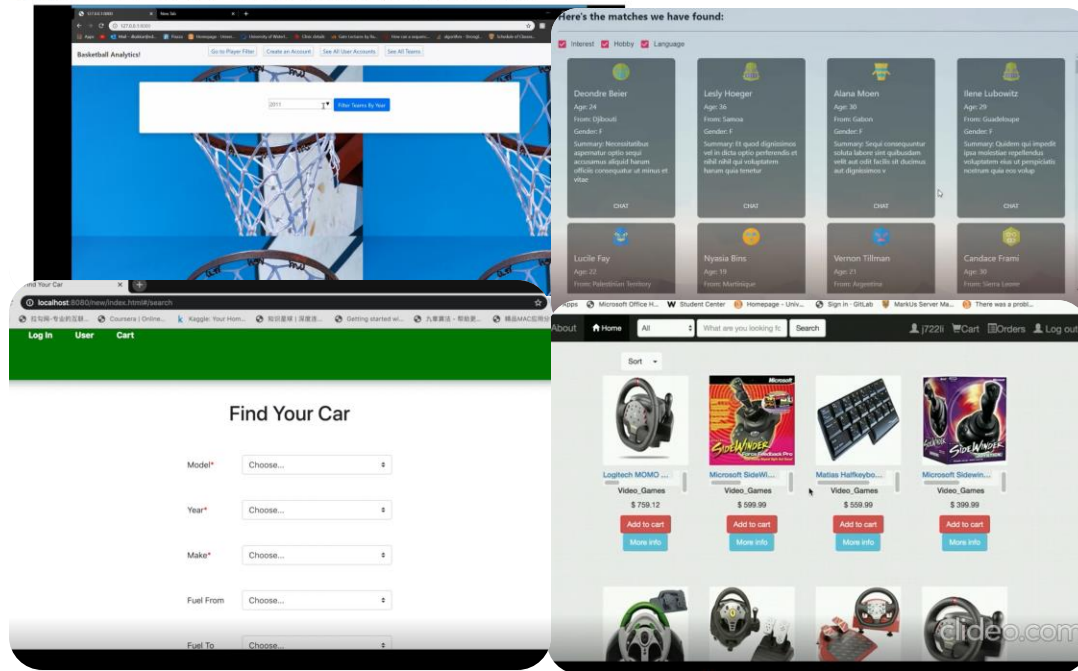
Project: DB-supported applications

- Team of 4-5 students (minimum 4, maximum 5)
- Project Timeline
 - Milestone 0: form a team by Thu, May 22
 - Milestone 1: proposal by Thu, Jun 19 (Switch cutoff)
 - Milestone 2: mid-term report by Tue, Jul 8
 - Milestone 3: demo + report + code
 - Demo in Week 12: From Mon, Jul 21 to Thu, Jul 24
 - Report and code by Tue, Jul 29
- Instructions released on Learn
- Start to brainstorm and find your teammates!
 - Members from **001, 002 and 003** sections are allowed.
 - Piazza is a good place to find teammates.

Project

- [Project demos](#) from previous years

Video Demo for NBA Season Statistics





Your turn to be creative

What's next?

- Lecture 2: Relational model

